COMAL TODAY 20

Dot Images See page 12

COMAL Today
6041 Monona Drive
Madison, WI 53716

Merry Christmas! Two Christmas Tree programs by John Hayes, Jason Barton, David Sharp, and Larry Taylor of Pocahontas County High School, are on Today Disk #20.

Bulk Rate
U.S.Postage
Paid
Madison, WI
Permit 2981

C128 CP/M SOUND page 24

UNERASE FILES page 11

MYTHS AND REALITY page 4

TURTLE GRAPHICS page 14

DOT IMAGES page 12

DOUBLE PRECISION MATH page 18

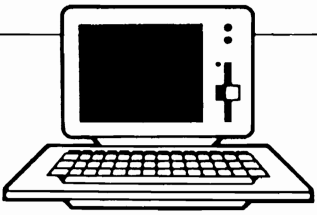
SHREDDER page 26

FILES page 21

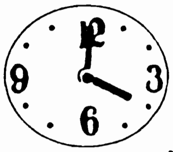
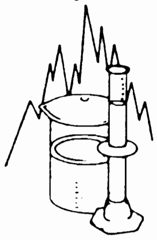
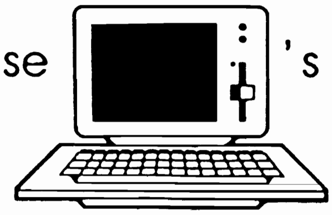
MOD TUTORIAL page 31

VOTE RESULTS page 7

BLACK BOX page 27

If you use a  **, you need**

The Computing Teacher

We publish articles from all over the in **The Computing Teacher** journal so that you'll get the best information. Information that's crystal clear, interesting ! and fun to use in the classroom. You can 1234 on us to have accurate, timely articles and to save you  and $ with our reviews and new product releases. We have columns for  , 1+3=4 , Logo, language arts and computers in the library. **The Computing Teacher**—for all those who use  's in the classroom.

ICCE, U of O, 1787 Agate Street, Eugene, OR 97403 USA

COMAL Today Issue 20 February 8, 1988

General

2 - Editor's Disk - Len Lindsay
3 - COMALites Unite - Richard Bain
4 - Myths and Reality - Len Lindsay
7 - Rumors - Captain *Buzz* COMAL
7 - Vote Results
8 - Letters
39 - Order Form

Beginners

6 - COMAL Clinic - Gary Franklin
21 - Files - David Stidolph
31 - Mod Tutorial - Bill Inhelder

Fun and Graphics

12 - Dot Images - Bill Inhelder
26 - Shredder - Luther & Dawn Hux
27 - Black Box - Brubaker & Zabel

2.0 Packages & Programming

14 - Turtle Graphics - Richard Bain
18 - Double Precision Math - Alan V Jones
24 - C128 CP/M Sound - Richard Bain
36 - Mandelbrot Revised - Robert Ross

Applications

11 - Unerase Files - Bob McCauley

Reference

5 - Best Books
32 - Disk Directories

Advertiser

ifc - Computing Teacher

Editor	Contributors	Contributors
Len Lindsay	Richard Bain	Carson Krol
	R L Brubaker	Len Lindsay
Assistants	Borge Christensen	Ed Matthews
Richard Bain	Captain COMAL	Robert Ross
Maria Lindsay	Gary Franklin	David Stidolph
David Stidolph	Brian Grainger	Chris Urholt
Geoffrey Turney	Dawn Hux	Richard Young
	Luther Hux	L W Zabel
Cover Artwork	Bill Inhelder	
Rhianon Lindsay	Alan Jones	

NOTICE: free subscriptions are terminated as of this issue. They may be renewed at our normal rates using the order form on page 39. Exceptions and questions should be referred to the Editor. Exceptions could include subscription exchange with newsletters that actually print COMAL information, and with schools who are actively using COMAL (USA only). Thank you.

Editor's Disk

As we enter our fifth year, and conclude our fourth super sale, we heard bad news. Since our sale offers many great deals, the word is out to wait for the next sale before buying anything. As you may guess, if this happens now (*the sale just ended*) it would be disastrous. Therefore, we are declaring a permanent sale. Check out the middle pull out order form to see for yourself. Yes, the free disks are still there, as are reduced prices on most things. Some prices are even lower than the previous sale prices.

Compiler: First CP/M users had a COMAL compiler. Then IBM. Now, C64 COMAL users have **Power Driver**, which includes a compiler along with 21 added commands to COMAL 0.14. For just $29.95 you get a complete **Power Box**, with about 250 ready to use procedures and functions, 3 disks full of utility programs, 3 books (*doc box style*), and a free copy of **Power Driver**. Even cartridge users can benefit from it: LIST your COMAL 2.0 program to disk; ENTER it into the **Power Driver**; SAVE the program to disk; compile it with the **Power Driver compiler**. This will work if your program uses only the keywords that are common with **Power Driver** (ie, no error trapping, import, loop, external, or packages).

Cartridge Special: If you don't have a C64 2.0 cartridge yet, we have a special for you! Eventually, you will want a copy of some of the books and disks out for the cartridge... so now get them all at the same time! **Cartridge Complete** gives you 11 books and 13 disk sides along with your cartridge ... all packed in a Doc Box. And it is specially priced at just $158.95 (limited to the 34 carts now in stock).

Questions: We no longer have a staff to answer your questions by phone. Two alternatives are: (1) Post your question on our Q&A message board on QLink... you should get a reply soon (2) Write to us and include a **stamped addressed envelope** for the reply (*and be patient, I am the only one here now to answer questions*).

Orders / Phone calls: If possible, mail in your orders. You still can use VISA/MC if you wish. I'll try to be available from 1pm - 4pm on Monday, Tuesday and Wednesday if you must place an order by phone. Finally, you can EMail your order to us on QLink. Include your name, address, charge card number and expiration date with your order. EMail it to **Captain C**.

QLink: this may be the best way to get your questions answered! Post them on our Q&A message board, or come to one of our two monthly meetings. Our COMAL section on QLink now has a new (and better) location:

- Go to CIN (Commodore Information Network)
  - Choose **Commodore Community SIGs**
    - Choose **Programmers Workshop**
      - Choose **COMAL**

We have our own upload/download libraries in our QLink section, as well as a conference room and Q&A board. Our meetings are held in our conference room at 10pm Eastern time every 1st Sunday and every **2nd** Thursday. See you there.

Program / Article submissions: We have changed our policy on submissions. We will send you a free copy of the newsletter and/or disk that it is published in. We won't send acknowledgement or disks until then.

Subscriptions: Our policy has always been that subscriptions are non-cancellable. Since most subscribers enjoy *COMAL Today* this hasn't been a problem. However, some users are upgrading their computer systems (usually to an Amiga) where there is no COMAL available. So, we now have a special offer on subscription cancelling. For each issue/disk cancelled, you may choose one backissue (*from the first column on page 39 order form*) **or** one disk (*from first column on page 40*). We also now limit subscriptions to 6 issues in advance. The size and format of the newsletter may vary, and the disks may be one or two sided. ■

COMALites Unite

by Richard Bain

David Stidolph and I gave a COMAL presentation at the UNCLE Commodore Users Group in Evanston, IL last night. We received a warm welcome from about forty computer enthusiasts. As with the other demonstrations we have given at computer meetings and shows, it was interesting to talk directly to the users to keep up to date with the current needs of the people actually using their computers.

Our presentation started with a brief talk on the history of COMAL. Then we showed off the COMAL 2.0 environment. We emphasized the **AUTO** command, keywords being listed in upper case, and the error messages. Anyone planning to give a COMAL presentation to their own users group should consider demonstrating how COMAL 2.0 prompts the user through the **FOR** loop. Start by typing a line number and the word **FOR**. COMAL responds with an error message saying you need a variable and places the cursor after the word **FOR**. Type in a variable name, **x** for example, and wait for the next error message. When you finally get to the ending value of the **FOR** loop on the CP/M or IBM systems, the line is relisted and the word **DO** is typed in for you (on the C64 you need to **LIST** the line again to see **DO**). By this time, you can expect to see half of the BASIC programmers in the audience drooling.

Our presentation continued with a demonstration of the Freeform Data Base program from *COMAL Today #12* and *The Amazing Adventures of Captain COMAL* fame. For flash, we showed off *draw'poker* from *COMAL Today #17*. Next, we switched to CP/M COMAL and again showed the data base program to prove that COMAL is compatible across versions. We concluded with the three dimensional airplane program from *COMAL Today #19*. This was the first time anyone in the audience had seen graphics from the CP/M mode of the C128.

Throughout our presentation, we answered questions from the audience. Two main themes of questions related to the high price of COMAL and when it would be available for the Amiga. Unfortunately, we could not give positive answers to either question. The price of COMAL 2.0 on the C64, C128, and IBM is not under our control and is not expected to go down any time soon. What we have seen of Mytech COMAL on the Amiga is no where near completion, although we hear another company is also currently working on Amiga COMAL.

One question we found very upsetting came from a user claiming that CP/M COMAL was worthless because CP/M is dead. Perhaps CP/M is not popular, but it is unfortunate that anyone will avoid what may be the best software for the money because of it.

The users groups have been very good for COMAL. As we have almost no budget for advertising and the major magazines have ignored COMAL, word of mouth via users groups is the primary route to spread COMAL. However, the users groups have not been able to support our company. By decentralizing the distribution of COMAL diskettes and support materials, there is simply not enough business to go around. This means we will have to cut back on our expenses and the amount of support we can give to COMAL users. David and I are moving on to other jobs (Len always had a second job). This does not mean we are abandoning COMAL, we like it too much to do that. It simply means we won't be able to work full time in COMAL. We won't be able to answer all your questions by phone or mail, and we can no longer exchange user group disks for your submissions. However, the COMAL Users Group will go on, our products will still be for sale, and David and I will still be contributing material for *COMAL Today*. ■

Myth & Reality

by Len Lindsay

As COMAL Users Group, U.S.A, Limited enters its fifth year, it may be time for reality to show through. While COMAL itself is doing quite well, we are not so fortunate. The top selling book sold only 11 copies in July. Only 7 new subscribers were added in April. This page includes other bits and pieces of reality.

Myth: COMAL Users Group has over 100,000 members. **Reality**: we have a bit more than 1,000 subscribers. April had 7 new subscribers, May 11, June 9, July 8, and August 9. That is only 44 subscribers in a 5 month period. Meanwhile, about 100 subscriptions expire **per month**. There **are** over 100,000 COMAL users, but less than 1% support *COMAL Today*.

Myth: With high prices on some COMAL items, we must be rolling in profits. **Reality**: many prices are beyond our control. We import some items from Denmark, and the de-valued dollar is worth only half of what it was worth a couple years ago. This doubles our cost to import the products. We sold the last batch of *Tutorial Binders* at less than our cost, just to honor the prices on our order forms! There is very little profit on the high priced items like the C64 Cartridge. We import them because they are the best, and we want them available to you.
Hint: get your cartridge now if you haven't yet. Only 34 in stock ... and the next batch will probably be significantly higher in price.

Myth: it is easy to just copy disks and books, and whip out a newsletter. **Reality**: we spend hours perfecting each new disk and book. It has taken three of us several months to polish the submitted material into one issue of *COMAL Today*. If users don't support our current efforts, there won't be money for new projects.

Myth: we have a large staff. **Reality**: we haven't had a secretary since May of 1986 and never have had a business manager or advertising manager. I fill those roles myself, along with editing the newsletter and books... in my spare time. I have a full time job as a Computer Operator running an IBM mainframe. We are sad to lose our only full time programmer and assistant editor (Richard Bain) and our part time programmer (David Stidolph). **They are two of the best programmers I have ever worked with!** It is a shame that our income no longer is enough to cover their wages. David continues work on Apple COMAL. Richard is moving to Minnesota, but promises to keep sending in material for *COMAL Today*.

Myth: with new COMAL implementations coming out, we must be expanding. **Reality**: the new implementations mean more computers (with incompatible disk formats). Until this month we had three small rooms. To save money we used doors laid across two drawer file cabinets as desks. Now we have only one small room. I am hoping for more new COMAL implementations, but I can't see how we can get MacIntosh, Amiga, and IBM PS/2 computer systems. The cost of each one of these systems would exceed our usual total monthly income of $3000-$5000. I didn't mention monthly profit, since for the past year there has been no profit at all.

Myth: by subscribing to *COMAL Today*, we become your personal programming consultants. **Reality**: while in the past we have tried to be helpful with your programming problems, this is no longer feasible. There no longer is a staff to answer the phone to discuss your applications. However, if you write to me **and include a stamped addressed envelope** I will try to reply. But be patient, please.

Myth: Authors of COMAL books make lots of money. **Reality**: Typical royalties for books are 15%. If a book sells only 4 copies per month (as did the #5 best selling books for October), and the price is $11, the total royalty for the month would be $6.60. Support our authors. Buy their books. They were published to help you! ■

Best Books

Last issue charted the top 10 best selling COMAL books of all time. The omitted monthly charts are included in this report for July through November.

Book sales have been very poor. Being an author myself, I feel sorry for those who spend hundreds of hours, working on a book, only to find a dozen people interested in it. For example, The #1 book for July sold only 11 copies! Support our COMAL authors!

New arrivals are Cartridge Keywords, CP/M COMAL Package Guide, and Common COMAL Reference. Plus, nearly all books are now printed as Doc Box style pages, including Cartridge Graphics & Sound, COMAL Quick & Utilities #2/#3, and even Superchip Notes.

It looks like COMAL 0.14 is obsolete. **Power Box** adds 21 new commands plus includes a free compiler ... along with its 3 books & 6 disks!

July 1987

#1 - COMAL Handbook
by Len Lindsay
#2 - Cartridge Graphics & Sound
by Captain COMAL's Friends
#3 - COMAL From A to Z
by Borge Christensen
#4 - Introduction to COMAL
by J William Leary
#5 - Graph Paper
by Garrett Hughes

August 1987

#1 - COMAL Handbook
by Len Lindsay
#2 - Introduction to COMAL
by J William Leary
#3 - COMAL Today - The Index
by Kevin Quiggle
#4 - Graph Paper
by Garrett Hughes
#5 - Cartridge Tutorial Binder
by Frank Bason & Leo Hojsholt

September 1987

#1 - COMAL From A to Z
by Borge Christensen
#2 - Introduction to COMAL
by J William Leary
#3 - COMAL Workbook
by Gordon Shigley
#4 - COMAL Handbook
by Len Lindsay
#5 - Packages Library #2
by various users
- Graph Paper
by Garrett Hughes

October 1987

#1 - COMAL 2.0 Packages
by Jesse Knight
#2 - Library of Functions & Procedures
by Kevin Quiggle
#3 - COMAL Handbook
by Len Lindsay
#4 - COMAL Quick & Utilities #2/#3
by Jesse Knight
#5 - Foundations with COMAL
by John Kelly
- COMAL Collage
by Frank & Melody Tymon
- Packages Library
by David Stidolph
- Packages Library #2
by various users
- COMAL Today - The Index
by Kevin Quiggle

November 1987

#1 - Library of Functions & Procedures
by Kevin Quiggle
#2 - COMAL Quick & Utilities #2/#3
by Jesse Knight
#3 - COMAL Today - The Index
by Kevin Quiggle
#4 - COMAL From A to Z
by Borge Christensen
#5 - Packages Library #2
by various users ■

COMAL Clinic - IF

by Gary Franklin

What is the difference between the following two programs?

```
INPUT "How many hamburgers did you eat? ":n
IF n<3 THEN
  PRINT "Are you still hungry?"
ELIF n>=3 THEN
  PRINT "Did you save any for me?"
ELIF n>5 THEN
  PRINT "I'll tell the doctor"
  PRINT "to pump your stomach."
ENDIF
```

------------------ and ------------------

```
INPUT "How many hamburgers did you eat? ":n
IF n<3 THEN
  PRINT "Are you still hungry?"
ELIF n>=3 THEN
  PRINT "Did you save any for me?"
ENDIF
IF n>5 THEN
  PRINT "I'll tell the doctor"
  PRINT "to pump your stomach."
ENDIF
```

The answer is that the first program has one **IF** structure and the second program has two **IF** structures. But what does this mean when you run the program? Let's run the first one (the underlined text must be typed by the user and the normal text is supplied by the computer):

```
RUN
How many hamburgers did you eat? 2
Are you still hungry?

RUN
How many hamburgers did you eat? 4
Did you save any for me?

RUN
How many hamburgers did you eat? 10
Did you save any for me?
```

Note, the third time you ran the first program, it did not call the doctor even though you ate more than five hamburgers. We will get back to that soon. For now, lets run the second program using the same numbers:

```
RUN
How many hamburgers did you eat? 2
Are you still hungry?

RUN
How many hamburgers did you eat? 4
Did you save any for me?

RUN
How many hamburgers did you eat? 10
Did you save any for me?
I'll tell the doctor
to pump your stomach.
```

When running the second program, the computer called the doctor after you ate ten hamburgers. The reason for the different output from the two programs must be the in the **IF** structure, but why? Both programs seem to check if you ate more than five hamburgers, but only the second one responds to that information.

Now that we have asked the right question, the answer is simple. In a multi-line **IF** structure, only one section of code can be executed. (In this example, a section of code refers to statements which are indented between other statements which are not indented. Therefore, the first program has three separate sections of code: the first two sections each consist of a single **PRINT** statement, and the third section is made of two **PRINT** statements.) The comparisons are made by the computer in the same order they are listed in the program. (The comparisons occur in the lines starting with **IF** or **ELIF**.) If the result of a comparison is **TRUE**, the section of code below that comparison is executed and then the programs skips to the end of the **IF** structure, specifically the line containing the word **ENDIF**.

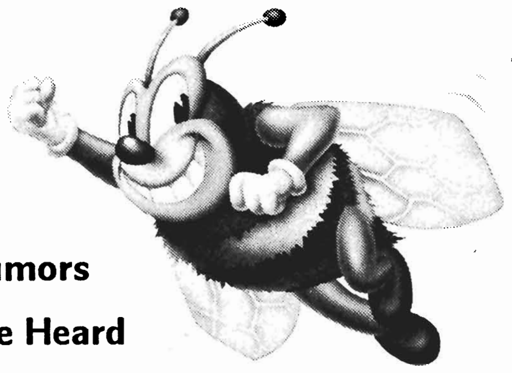
more»

Therefore, the first time you ran the first program, it realized that you ate less than three hamburgers and asked if you were still hungry. It didn't bother to check the other two conditions because it was satisfied with the first result. The second time you ran the program it realized that you ate at least three hamburgers, so instead of asking if you were still hungry, it asked if you saved any hamburgers. It didn't need to check if you ate more than 5 hamburgers because it already had a **TRUE** result. The third time you ran the first program it responded exactly the way it did the second time. It didn't matter that you had eaten more than five hamburgers because COMAL skipped over the last condition after discovering you ate at least three hamburgers.

The second program split the **IF** statement from the first program into two separate **IF** statements. As you now know, this was necessary. It prevented COMAL from skipping over the comparison for eating five hamburgers, thus allowing you to receive the necessary medical attention.

Now that you know that only one section of the **IF** structure can be executed during any given program, you are ready to learn a little trick. It allows you to offer medical attention to someone who eats more than five hamburgers without first asking if any hamburgers are left. Why be cruel to someone with a tummy ache?

```
INPUT "How many hamburgers did you eat? ":n
IF n<3 THEN
  PRINT "Are you still hungry?"
ELIF n>5 THEN
  PRINT "I'll tell the doctor"
  PRINT "to pump your stomach."
ELIF n>=3 THEN
  PRINT "Did you save any for me?"
ENDIF
```
■



Rumors We Heard

by Captain "*Buzz*" COMAL

While we haven't heard from Mytech for months, there are rumblings that another Amiga COMAL might be in the works (great since that is the number one computer that the **vote** results below show users wish there was a COMAL for). Watch QLink for rumor updates!

UniComal is putting the final touches on their latest implementation... IBM PS/2 COMAL 2.2 with VGA graphics. They promised us a review copy ... and we know someone with PS/2 computers that we can test it on (we can't afford to buy a PS/2 system ourselves).

VOTE

Your favorite article in *COMAL Today #19*:

#1 - Rotating 3D Image - page 53
#2 - COMAL 0.14 Power Driver - page 12
#3 - COMAL Coloring Book - page 45
#4 - Sample Book Pages - page 16

What COMAL do you use?

C64 0.14	■■■■■■■■■■■■■■■■■■■■■■■■■■■
C64 cart	■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
Superchip	■■■■■■■■■■■■■■■■■
CP/M	■■■■■■■■
IBM	■■

What computer should COMAL next appear for?

Amiga	■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
Apple II	■■■■■■
Atari ST	■■■■
$100 IBM	■■■■

Letters

Report From England

Dear Len - The only commercial COMAL product in the UK, apart from UniComal, is Acornsoft COMAL. This has sold reasonably well. It is interesting that COMAL is mentioned heavily in the advertising of the latest Acorn/ Olivetti machine (Archimedes). This is a RISC machine mainly targeted at Education but I do not know what COMAL they are talking about.

The Standards meeting was quite interesting this time. A lot of work was done and important work too. The MODULE proposals were accepted which is probably the most important outcome. A timescale has been set for getting a printed copy of the standard out. You, I and everybody else is frustrated by the fact that we cannot view the current standard! Most of my proposals on Graphics were accepted. A number of your proposals were also accepted; others were not accepted but in many cases this was due to the fact that they were not written in Standard "*language*". For example, cross referring to the extensions in the kernal standard was not considered necessary since the extensions stand on their own. It is a pity you could not come to hear the discussions but it is a long way! The one disappointing feature of the meeting was the lack of agreement of the null separator. It looks as if this is never going to be solved. I must admit I can see both sides of the argument. If this had been thought about at the beginning I am sure we would not have a problem now. - Brian Grainger, UK

Schools / Borge Christensen

Dear Sir: Thank you for promptly sending me some material for my "COMAL Drive". So far I have done a presentation for the University of Washington CUG and the Northwest CUG. There are three more CUGs in the area that I'll offer my "COMAL Presentation" to before Thanksgiving.

I am doing this because I like COMAL and know how much easier it is to learn compared to FORTH, PASCAL, and BASIC, which I struggled with for quite some time before I found relief with COMAL.

About three years ago I wrote to a Danish firm, (whose address I now have lost!) and ordered *Beginning COMAL* and COMAL cartridges for my grandsons in Norway. I told them to bill my bank in Norway. A week later somebody called from Denmark about six in the morning, and when my wife said I wasn't up yet, the caller apologized for not realizing that it's three hours earlier here than on the East Coast and said he would call later. So he did, and it was none less than Professor Christensen! I was speechless! "*I can send COMAL 0.14 on disk to your grandsons*" he said, and when I said they didn't have disk drives yet, he said: "*men sa sender jeg band til guttene!*". (Well, then I'll send the boys tapes.) - So he did, and the bill to my bank was very reasonable with no charge for the two telephone calls! Can you beat that? It was like having talked to an old friend. - Chris Urholt, Bellevue, WA

We did manage to get Borge Christensen to come to the MARCA show in Pennsylvania in 1985. He truly is a wonder! COMAL is his creation ... and to think he did it over 10 years ago. Other languages are finally starting to catch up - but COMAL still leads the pack.

Canada Schools & The ICON

Dear Captain COMAL - I attended a Commodore Club meeting and had the pleasure of hearing Kevin Quiggle give a presentation on COMAL. That was about two years ago now. I was so impressed by COMAL that I began to teach COMAL to my senior students and have enjoyed the language ever since.

I was pleased to see your specials in *COMAL Today #18*. It will be nice to be able to get

more»

back issues of *COMAL Today* and the COMAL books at reduced rates. Maybe in the future there could be a school special on *COMAL Yesterday* too. (*Sorry, but as it is, our cost is nearly the same as its selling price.*)

It is not completely due to lack of funds that we do not use the 2.0 cartridges, (it is difficult in a classroom setting to secure the cartridge); therefore, we must rely on the 0.14 COMAL disk. I find it awkward with no built-in VAL and STR$ functions as in Commodore Basic. I work on a network with eight C64's and am unable to use the disk drive version (of the STR$ and VAL functions) that appeared in *COMAL Today*. Kevin Quiggle modified and I adapted the value-string routine that appears in his *Library of Functions & Procedures* book. It works, but I would think by now that the VAL and STR$ functions would be included in the main COMAL 0.14 language as it is in COMAL 2.0. (*Power Driver has both STR$ and VAL along with 19 other added commands. It is disk loaded like 0.14 so it should meet your needs.*)

The following situation is more critical. The Ministry of Education of Ontario has directed that Structured Programming be taught. This is great for COMAL because it is not an "add-on" structured language like most of the languages currently available, but a pure structured language. The problem is with the computer hardware. The Ministry of Education greatly influences computer purchases by provincial grants available to school boards. Currently a school board can purchase a 512K **ICON** computer (about $5,000) which is networked to a 64-meg fileserver for about the same price as a Commodore 64 system at retail price (the C64 does not have provincial approval). Now the schools are only purchasing **ICON**s. The only true structured languages available on the Icon are Pascal and C, both of which are more difficult than COMAL. If there were COMAL for the ICON, I feel that many more teachers would be more than willing to teach COMAL. This market is very large; all Ontario high schools and an increasing number of grade schools as they enter the computer age. The ICONs are made by UNISYS and are presently running on an operating system called QNX version 2.25, which I am told, is a version if UNIX. This could be a good project for COMAL developers and a good boost for the COMAL language.

If anyone would like more information, I would be only too glad to help because I can see COMAL's potential in education and also its demise in Ontario if it cannot run on the ICON. I hope someone sees the potential and the opportunity for COMAL to penetrate the market "north of the border". - Carson Krol, RR #2 Manning Road, Maidstone, Ontario N0R 1K0

COMAL At The U

Dear Sir: I have been enjoying and using COMAL with my C64 for some time now. I am currently a graduate student at the University of Georgia working on my masters degree in computer based education technology. I was surprised to discover that, although all of my professors are familiar with computers and the various high level languages, none had heard of COMAL. This was particularly surprising when one of my courses is programming in Pascal. I know that your budget is probably pretty tight but is there a way I could borrow a copy of IBM COMAL or get a demo? I think that my professors and fellow students would be impressed. In my Pascal class I am constantly asking questions about how Pascal can be used to various things, and keep getting the response that you can't do that with Pascal. COMAL, of course, does it easily. - Richard Young, Georgia

Sorry Richard, but UniComal does not allow review copies of their IBM COMAL, nor will they permit us to create a Demo disk for it. Perhaps once your professors see it run on your C64 with the knowledge that the IBM version is even better, they would get the IBM COMAL.

more»

Super Chip vs. $15,000 System

Dear Mr. Lindsay: Recently a friend asked me to write a program for communicating via RS-232 with some computer-controlled industrial machines. For reasons of cost and ease of programming, I chose a C64 with COMAL 2.0 and Super Chip.

The application required hard-copy capability, but there would be times when a printer would not be connected to the computer. In addition, the system would frequently be used by people who were not regular users of computer systems. So I decided it would be nice to avoid the *device not present* crash that would result if the user selected the printer option when the printer was not connected, and give a way out that would not destroy the data in the memory of the computer.

Attached is a listing of PROC print'out that does the printing after calling FUNC printer'ready to check if the printer is there. If the printer is not present, the error is TRAPped, an explanation is printed on the screen, and the user is given the choice of cancelling the hardcopy request.

FUNC get'reply$(string$) is called to obtain the user input indicating whether to cancel. If you tell it not to cancel, it repeats testing for the printer until the printer is present or you do tell it to cancel. Thanks to David Stidolph for this latter FUNCtion and for getting me to start writing FUNCtions when they are more appropriate than PROCedures.

Note that FUNC printer'ready prints a blank line if the printer is ready. This was not a problem in this application.

Incidentally, the friend is quite impressed with the capabilities of the program and of COMAL and the Super Chip; that the system offered by the manufacturer of the machines for this purpose does less and costs $15,000 may be a factor. - Ed Matthews, Springfield, MO [*Note: the following is not a complete program - only the parts mentioned in the letter so you may see what he is referring to.*]

```
USE system
USE files
PROC print'out
  IF printer'ready THEN
    SELECT OUTPUT "lp:"
    FOR block'number:=0 TO total'blocks DO
      PRINT block$(block'number)
    ENDFOR block'number
  ENDIF
  SELECT OUTPUT "ds:"
ENDPROC print'out
//
FUNC printer'ready
  LOOP
    TRAP
      SELECT OUTPUT "lp:"
      PRINT
      SELECT OUTPUT "ds:"
      RETURN TRUE
    HANDLER
      SELECT OUTPUT "ds:"
      PRINT ""147""17""17" The printer is not
      ready to print." // wrap line
      PRINT ""17"   Please turn on the printer
      and" // wrap line
      PRINT ""17"   prepare it to print."
      PRINT ""17""17"   Cancel printing?
      ("18"Y"146" or "18"N"146")"; // wrap line
      IF inkey$ IN "Yy" THEN RETURN FALSE
    ENDTRAP
  ENDLOOP
ENDFUNC printer'ready
//
FUNC get'reply$(string$)
  REPEAT
    reply$:=inkey$
  UNTIL reply$ IN string$
  PRINT reply$
  RETURN reply$
ENDFUNC get'reply$ ■
```

Unerase Files

by Bob McCauley

Suppose you just mistakenly entered

DELETE "favorite prog"

Or, infinitely worse, you entered

DELETE "*favorite prog"

In the second case CBM DOS ignores everything after the "*" and deletes **every** file on the disk. The old solution was to grab your favorite disk editor (Phyrne Bacon's *disk'editor* is in *COMAL Today #13*). However, mucking about in the disk directory is not for the faint of heart.

Chip.unerase is a COMAL 2.0 program on *Today Disk #20* which requires Super Chip and takes you step-by-step through the process of examining the disk for erased files, selecting the file, checking if it **can** be UnErased, and UnErasing it. Several hints are appropriate:

1) When you (accidentally) erase something you didn't want to erase, **stop immediately** and do not perform any **SAVE** or **WRITE** operations to that disk.

2) If you forget hint #1 or want to recover something deleted a while back, it is possible the file name you want is duplicated by either an active file, a deleted file(s), or both. *File'unerase* lets you **RENAME** the active file if desired. If you have multiple erased copies you need to pick and choose the one you want to UnErase. If you UnErase the wrong one, no big deal! Just **DELETE** it and go back to UnErase another one.

3) If you have UnErased any files you **must** Validate the disk before using it. Otherwise the Block Allocation Map (the BAM) will not protect the newly UnErased file and the next **SAVE** could overwrite it.

4) Whenever you are fooling around in the directory area, even with a (supposedly) checked out, bug-free program (there ain't any), **be careful.**

A Tutorial on Unerasing Files
(For Commodore 1541/CBM DOS)

Let's see what happens when we **SAVE** a file. The first thing DOS does is to look in the Block Allocation Map (the BAM, where DOS records whether a block at a particular track/sector (t/s) is vacant or not). It then saves the file, picking vacant t/s along its path according to its own arcane rules. Each block is linked to the next one by recording the **next** t/s in the first two bytes of **this** block. When all is saved and the blocks are counted, the DOS writes the updated BAM to track 18, sector 0 and then builds a new directory entry in the **first** vacant entry location it finds. It writes the name of the file, the length, the starting t/s, and, most important for UnErasing, the file type (eg: sequential, program, user, or relative). It stores the file type as a one byte character, with a CHR$(0) used to represent a deleted file.

When DOS deletes a file, the process is a bit simpler. First, DOS goes to the directory and looks for the starting t/s. It then goes to that block and, following the trail marked by the first two bytes in each block, traces the entire file. It de-allocates each block in the BAM and then changes the file type in the directory entry to a '0'. The file itself remains intact. To UnErase this file you need do only three things:

1) Check the file itself to see if it still contains the same number of blocks. If it does you have a good bet (though not 100%) that it is still the same file.
2) Change the file type byte to the proper one denoting the type file you are dealing with. See your Disk Drive Manual for details.
3) Rebuild the BAM. Just **PASS "V0"** ■

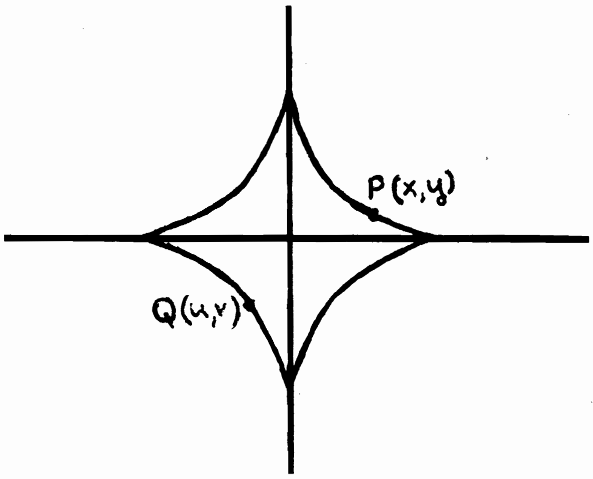
Dot Images

by Bill Inhelder

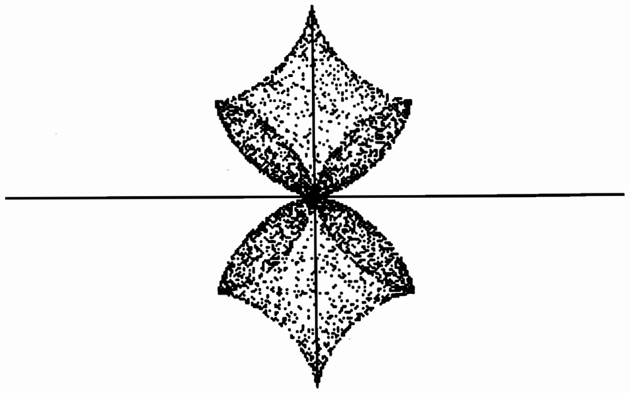
Dot'images on *Today Disk #20* is a program which uses equations in a unique way to produce designs composed of dots. The density of the dots can produce designs which are superimposed upon another design of lighter shading. The search for equations (mathematical functions and relations) which create interesting images can be quite a challenge. Eight such equations are offered in the program for the user's selection.

The position of each point plotted on the screen is determined by randomly selecting a point on the right hand portion of the function or relation (the first and fourth quadrants) and adding its coordinates to the coordinates of a second point randomly selected from the left hand portion of the function or relation (the second and third quadrants).

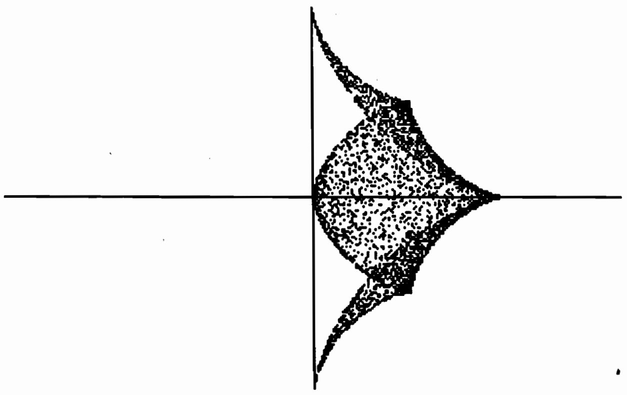
Thus if the graph of the hypocycloid of 4 cusps is used to generate a design, point P(x,y) is randomly selected on the right side and point Q(u,v) on the left.



Point T(x+u,y+v) which represents the sum of P and Q is plotted to the screen. Repeating this process 5000 times creates the following design.



If point T(x-u,y-v) is determined by subtracting Q from P, the following design is produced.



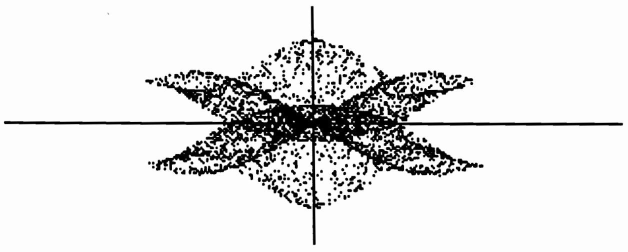
The addition or subtraction forms of the equations in the program are indicated by the suffixes sum and dif and may be selected by the user.

It is interesting to watch the patterns develop as the points are plotted. Usually the full beauty of the pattern is not apparent until most of the 5000 points are plotted. Depending on the nature of the equation, the process might take 10 to 20 minutes. After each design is complete, press any key to return to the menu. Relations which possess both horizontal and vertical symmetry tend to produce the most interesting designs.

more»

The following program can be used to produce the density images above. The *dot'images* program on *Today Disk #20* offers several other equations to produce a wider variety of pictures.

```
PAGE
USE graphics
PRINT AT 12,7: "DENSITY - A mathematically"
PRINT AT 13,10: "determined dot image"
PRINT AT 18,22: "By Bill Inhelder"
FOR i:=1 TO 2000 DO NULL
PAGE
LOOP
  options
  graphicscreen(0)
  clearscreen
  background(1)
  pencolor(0)
  k:=1.3
  axes
  CASE choice OF
  WHEN 1
    hypocycloid'sum
  WHEN 2
    hypocycloid'dif
  OTHERWISE
    END
  ENDCASE
  WHILE KEY$="" DO NULL
  textscreen
ENDLOOP
//
PROC options
  PRINT AT 4,1: "The following functions or
  relations" // wrap line
  PRINT "are available:"
  PRINT
  PRINT " 0. quit program"
  PRINT " 1. hypocycloid of 4 cusps-sum"
  PRINT " 2. hypocycloid of 4 cusps-difference"
  PRINT
  INPUT "Enter the desired number :": choice
  IF choice=0 THEN END
  PAGE
ENDPROC options
//
PROC axes
  moveto(160,0)
  drawto(160,199)
  moveto(0,100)
  drawto(319,100)
ENDPROC axes
//
PROC hypocycloid'sum
  FOR i:=1 TO 5000 DO
    a:=PI/2*RND
    IF RND>.5 THEN a:=-a
    xr:=50*(COS(a))^3
    yr:=50*(SIN(a))^3
    b:=PI/2+PI*RND
    xl:=50*(COS(b))^3
    yl:=50*(SIN(b))^3
    x:=xr+xl+160
    y:=100+yr+yl
    plot(x,y)
  ENDFOR i
ENDPROC hypocycloid'sum
//
PROC hypocycloid'dif
  FOR i:=1 TO 5000 DO
    a:=PI/2*RND
    IF RND>.5 THEN a:=-a
    xr:=50*(COS(a))^3
    yr:=50*(SIN(a))^3
    b:=PI/2+PI*RND
    xl:=50*(COS(b))^3
    yl:=50*(SIN(b))^3
    x:=xr-xl+160
    y:=100+yr-yl
    plot(x,y)
  ENDFOR i
ENDPROC hypocycloid'dif
```



■

Turtle Graphics

by Richard Bain

The C128 CP/M graphics package has commands to plot points and draw lines on the monocolor 640X200 double hi-res graphics screen. We felt you might be interested in problems involved in putting the package together and in the algorithms for the commands. *[Note: the COMAL algorithms presented below do not represent a working program. You will not be able to run them on any system and they are not included on disk. However, they are educational and let you know what you get with the package.]*

The graphics package is not the result of any one person's work. When we were given the rights to distribute CP/M COMAL, we were also given also given a preliminary source code file for a graphics package. It had a routine to draw a line from the current (X,Y) coordinate to another coordinate. The code was organized in such a way as to allow it work on any CP/M machine which supported graphics. The line drawing routine calculated the (X,Y) coordinates of the points to plot. The plot a point routine set the pixel in the video memory. The idea was to have a general line drawing algorithm that works on all CP/M computers plus a point plotting algorithm specific to the computer it is used on. This way, the entire graphics package could be ported to other computers by only changing the point plotting routine. Unfortunately, we were not given that routine.

Ray Carter, one of our original CP/M COMAL testers, quickly came up with the routine necessary to plot a point on the graphics screen. In doing this, we rediscovered a problem previously encountered with the **C128** package for the COMAL cartridge. There are several incompatible versions of the C128 VDC video chip. A small fix bypassed part of the problem, but another problem remained. A 640X200 graphics screen requires 16,000 bytes of VDC memory. The chip which comes with the C128 has 16,384 bytes. As this chip must also store the text screen and the character set, there is a small problem. However, the chip can be modified to hold 64K of RAM. This is enough to store both the text and the graphics screens.

In order to use graphics with the standard 16K chips, the character set must be saved before the graphics screen is activated. Then the character set must be copied back to the VDC chip before the text screen can be re-entered. This presented a major stumbling block, but the result is that we are releasing a preliminary C128 CP/M COMAL **Font** package along with the **Graphics** package. Note, there is not enough memory to store the graphics picture while the textscreen is being displayed. Therefore, the picture on the graphics screen is lost each time you leave the graphics screen to enter the textscreen.

The COMAL graphics routines below require five machine language support routines which are very machine specific. Only **clearscreen** is available to the COMAL programmer.

- **Setgraphic** - displays the graphics screen.
- **Settext** - displays the text screen.
- **Clearscreen** - erases the graphics screen.
- **Setpoint(x,y)** - usually plots a pixel at the (X,Y) coordinate. If the **penup** command has been issued or the point is outside the viewport, this command does nothing. If the **penstate** is (0) this command erases the pixel and if the penstate is (-1) this command flips the pixel. **Setpoint** uses the default screen coordinates and is not affected by the **window** command.
- **Getpoint(x,y)** - is a function which returns **TRUE** if the pixel at the (X,Y) coordinate is set, **FALSE** if the pixel is clear, or **(-1)** if the point is outside of the current viewport. **Getpoint** uses the default screen coordinates and is not affected by the window command.

The COMAL functions **absx** and **absy** (listed below) are used to convert (X,Y) coordinates

more»

from the window coordinates (used by the COMAL programmer) to the absolute (X,Y) screen coordinates (needed by the machine language point plotting routines). Absx and Absy are not available to the COMAL programmer.

The following variables are used by the graphics package and can be called as functions by the user: **xcor, ycor, heading, wxmin, wymin, wxmax, wymax, vxmin, vymin, vxmax, vymax.**

The following internal variables used by the graphics package. They save time, but cannot be accessed by the user: **xscale, xoffset, yscale, yoffset,** and **penstate.**

Now, for the COMAL routines which were used as a guide for the commands in the C128 CP/M Graphics Package.

```
PROC graphicscreen
  setgraphic
  vxmin:=0; vxmax:=639
  vymin:=0; vymax:=199
  window(0,639,0,199)
  pen(TRUE)
  home
  clearscreen
ENDPROC graphicscreen
//
PROC textscreen
  settext
  PAGE
ENDPROC textscreen
//
PROC moveto(x,y)
  xcor:=x; ycor:=y
ENDPROC moveto
//
PROC move(x,y)
  moveto(xcor+x,ycor+y)
ENDPROC move
//
PROC setheading(degrees)
  heading:=((degrees MOD 360)+360) MOD 360
ENDPROC setheading
//
PROC right(degrees)
  setheading(heading+degrees)
ENDPROC right
//
PROC left(degrees)
  setheading(heading-degrees)
ENDPROC left
//
PROC home
  moveto(0,0)
  setheading(0)
ENDPROC home
//
PROC window(xmin,xmax,ymin,ymax)
  IF xmin>=xmax THEN REPORT
  IF ymin>=ymax THEN REPORT
  // the lines below are only executed
  // if the parameters are valid
  wxmin:=xmin; wxmax:=xmax
  wymin:=ymin; wymax:=ymax
  xscale:=(vxmax-vxmin)/(wxmax-wxmin)
  xoffset:=vxmin-(wxmin*xscale)
  yscale:=(vymax-vymin)/(wymax-wymin)
  yoffset:=vymin-(wymax-wymin)
ENDPROC window
//
PROC viewport(xmin,xmax,ymin,ymax)
  IF xmin>=xmax THEN REPORT
  IF ymin>=ymax THEN REPORT
  IF xmin<0 THEN REPORT
  IF ymin<0 THEN REPORT
  IF xmax>639 THEN REPORT
  IF ymax>199 THEN REPORT
  // the lines below are only executed
  // if the parameters are valid
  vxmin=xmin; vymin=ymin
  vxmax=xmax; vymax=ymax
  window(wxmin,wxmax,wymin,wymax)
  // resetting the window changes the variables
  // xscale, xoffset, yscale, and yoffset
  // the window 'follows' the viewport
ENDPROC viewport
//
PROC plot(x,y)
  setpoint(absx(x),absy(y))
```

more»

```
ENDPROC plot
//
FUNC getpixel(x,y)
  RETURN getpoint(absx(x),absy(y))
ENDFUNC getpixel
//
PROC line(x1,y1,x2,y2)
  x1a:=absx(x1)
  x2a:=absx(x2)
  y1a:=absy(y1)
  y2a:=absy(y2)
  IF x1a<vxmin OR x1a>vxmax THEN RETURN
  IF x2a<vxmin OR x2a>vxmax THEN RETURN
  IF y1a<vymin OR y1a>vymax THEN RETURN
  IF y2a<vymin OR y2a>vymax THEN RETURN
  // the line is only drawn if the entire
  // line is within the viewport
  deltax:=ABS(x2a-x1a); deltay:=ABS(y2a-y1a)
  xdir:=SGN(x2a-x1a); ydir:=SGN(y2a-y1a)
  xline:=x1a; yline:=y1a
  setpoint(xline,yline)
  IF deltax>=deltay THEN
    error:=2*deltay-deltax
    errorplus:=2*(deltay-deltax)
    errorminus:=2*deltay
    FOR count:=1 TO deltax DO
      IF error>0 THEN
        yline:+ydir; error:+errorplus
      ELSE
        error:+errorminus
      ENDIF
      xline:+xdir
      setpoint(xline,yline)
    ENDFOR count
  ELSE
    error:=2*deltax-deltay
    errorplus:=2*(deltax-deltay)
    errorminus:=2*deltax
    FOR count:=1 TO deltay DO
      IF error>0 THEN
        xline:+xdir; error:+errorplus
      ELSE
        error:+errorminus
      ENDIF
      yline:+ydir
      setpoint(xline,yline)
    ENDFOR count
  ENDIF
ENDPROC line
//
PROC drawto(x,y)
  line(xcor,ycor,x,y)
  moveto(x,y)
ENDPROC drawto
//
PROC draw(x,y)
  drawto(xcor+x,ycor+y)
ENDPROC draw
//
PROC forward(d)
  draw(d*SIN(heading),d*COS(heading))
ENDPROC forward
//
PROC back(x)
  forward(-x)
ENDPROC back
//
PROC circle(x'center,y'center,radius)
  ellipse(x'center,y'center,radius,radius)
ENDPROC circle
//
PROC ellipse(x'center,y'center,x'radius,y'radius)
  arcs:=64
  dt:=2*PI/arcs; d2t:=dt*dt/2
  d3t:=d2t*dt/3; d4t:=d3t*dt/4
  even:=1-d2t+d4t; odd:=dt-d3t
  x'odd:=odd*x'radius/y'radius
  y'odd:=odd*y'radius/x'radius
  xcircle:=x'radius; ycircle:=0
  moveto(xcircle+x'center,ycircle+y'center)
  FOR t:=1 TO arcs DO
    xcircle0:=xcircle
    xcircle:=xcircle*even-ycircle*x'odd
    ycircle:=ycircle*even+xcircle0*y'odd
    drawto(x'center+xcircle,y'center+ycircle)
  ENDFOR t
ENDPROC ellipse
//
PROC pen(state)
  penstate:=state
ENDPROC pen
//
```

more»

```
FUNC absx(x)
  RETURN x*xscale+xoffset
ENDFUNC absx
//
FUNC absy(y)
  RETURN y*yscale+yoffset
ENDFUNC absy
```

Some of the most important routines in the graphics package don't change the picture on the graphics screen. The **viewport** command is used to restrict drawing to a limited portion of the screen. The **window** command changes the coordinates used to plot points within the viewport. **Penup** and **Pendown** (not listed above) are used to disable and re-enable the drawing commands. The **pen(penstate)** command alters the meaning of the drawing commands. If **penstate** is **TRUE** (positive) the drawing commands will set points, if **penstate** is **FALSE** (0) the drawing commands will erase points, and if **penstate** is **(-1)** (negative) the drawing commands will flip the points.

The **line** command is one of the most complicated routines in the package. The first thing it does is convert the endpoints of the line from the coordinate system set by the **window** command to the coordinate system used by VDC memory. This step allows integer math to be used on all the points in between, saving time compared to using real math. Then it determines if the second point is above, below, to the right or to the left of the first point. It also checks if the line is closer to being vertical or horizontal. Based on this, it can move one point at a time in the long direction, and move either zero or one point in the short direction until the endpoint is reached. **Line** does not change **xcor**, **ycor** or **heading**.

The **circle** and **ellipse** commands break the shape down into several small arcs. A straight line is used to approximate the arcs. Due to nature of the screen coordinate system, the **ellipse** command is more likely to be used to draw a round circle than the **circle** command.

The **fill** command could not be included in the graphics package, although a COMAL **fill** procedure is include on the graphics package disk. It is too complicated to explain in this article even though it started as a nine line procedure:

```
PROC fill (x,y)
  WHILE NOT getpixel(x,y) DO
    PLOT (x,y)
    fill (x-1,y)
    fill (x+1,y)
    fill (x,y-1)
    fill (x,y+1)
  ENDWHILE
ENDPROC fill
```

This highly recursive procedure works great, but it requires a mainframe's worth of memory to run it. The algorithm used in the **fill** procedure on the graphics disk is also recursive, but it can fill any convex polygon before needing recursion. However, there are some shapes which do require enough recursion to cause memory problems.

Plottext is also included in the graphics package. It is limited to byte boundaries. This means that plotting a character to (0,0) or (1,0) will actually plot the character to the same screen position, namely (0,0). ■

Double Precision Math

by Alan V. Jones

COMAL does not have double precision variables or math. In the course of developing and testing numerical algorithms with COMAL I often wished that I had double precision math available. This would allow easier error analysis, and iterative improvement techniques. The procedures presented here effectively provide that capability.

The first step is to determine the normal floating point number arithmetic characteristics. Reference 1 gives a FORTRAN routine for determining the base, number of base digits, and whether rounding or chopping is done. I translated this into a COMAL procedure which can be used by any computer running COMAL. For the C64 with COMAL 2.0 the results are: base 2, 32 digits of base 2 precision, and proper rounding. This is good math. A more common constant is EPSILON or MACHEPS which is the machine floating point precision. It is defined as the smallest number that can be added to 1.0 with the result not equal to 1.0. MACHEPS for the C64 is 1/(2^32) and 1/(beta#^t#) for any machine. MACHEPS is usually computed more directly from a simpler routine.

```
PROC environ(REF beta#,REF t#,REF round#)
CLOSED // wrap line
  round#:=1
  a:=2; b:=2
  WHILE (a+1)-a=1 DO
    a:=2*a
  ENDWHILE
  WHILE a+b=a DO b:=2*b
  beta#:=(a+b)-a
  temp:=beta#-1
  IF a+temp=a THEN round#:=0
  t#:=0; a:=1
  temp:=beta#
  REPEAT
    t#:+1; a:=a*temp
  UNTIL (a+1)-a<>1
ENDPROC environ
```

We can now develop a set of double precision math routines that use only the existing floating point math. A double precision number A is represented as a+aa; two normal non overlapping floating point numbers. I found the double precision routines in reference 2 which is an extension of Dekker's work [3]. Linnainmaa's paper provides proofs for his double precision algorithms that are valid for almost any known arithmetic. He also gave Pascal routines. Since the C64 has such nice arithmetic, I simplified the routines to a state that should correspond to Dekker's algorithms.

```
PROC exactmul2(a,c,REF x,REF xx) CLOSED
  // x + xx = a*c
  //maxreal=1.70141183e38
  IMPORT constant,maxtest,scale2
  //:=2^16+1,maxreal/2^16,2^32
  uscaled#:=FALSE
  IF ABS(a)>maxtest THEN //overflow
    //                    protection
    a:=a/scale2
    scaled#:=TRUE
  ELIF ABS(c)>maxtest THEN
    c:=c/scale2
    scaled#:=TRUE
  ELSE
    scaled#:=FALSE
    IF ABS(a*c)<0 THEN //underflow protection
      uscaled#:=TRUE
      IF ABS(a)>ABS(c) THEN
        c:=c*scale2
      ELSE
        a:=a*scale2
      ENDIF
    ENDIF
  ENDIF
  t:=a*constant; a1:=(a-t)+t; a2:=a-a1
  t:=c*constant; c1:=(c-t)+t; c2:=c-c1
  x:=a*c
  xx:=(((a1*c1-x)+a1*c2)+c1*a2)+c2*a2
  IF scaled# THEN
    x:=x*scale2
    xx:=xx*scale2*scale2
  ELIF uscaled# THEN
```

more»

```
    x:=x/scale2
  ELSE
    xx:=xx*scale2
  ENDIF
ENDPROC exactmul2
//
PROC longmul(a,aa,c,cc,REF x,REF xx) CLOSED
  // x+xx:=(a+aa)*(c+cc)
  IMPORT exactmul2,scale2
  exactmul2(a,c,z,qq)
  IF ABS(z)>0 THEN //underflow protection
    zz:=((a*cc+aa*c)+qq)/scale2
    x:=z+zz
    xx:=((z-x)+zz)*scale2
  ELSE
    zz:=(a*cc+aa*c)+qq; x:=z*scal2+zz
    xx:=(z-x)+zz; x:=x/scale2
  ENDIF
ENDPROC longmul
//
PROC longdiv(a,aa,c,cc,REF x,REF xx) CLOSED
  IMPORT exactmul2,scale2
  //x+xx:=(a+aa)/(c+cc)
  z:=a/c
  exactmul2(c,z,q,qq)
  IF ABS(z)>0 THEN //underflow protection
    zz:=(((((a-q)*scale2-qq)+aa)-z*cc)/scale2)/c
    x:=z+zz
    xx:=((z-x)+zz)*scale2
  ELSE
    zz:=((((a-q)*scale2-qq)+aa)-z*cc)/c
    x:=z*scale2+zz
    xx:=(z-x)+zz
    x:=x/scale2
  ENDIF
ENDPROC longdiv
//
PROC longadd2(a,aa,c,cc,REF x,REF xx) CLOSED
  IMPORT scale2
  // x+xx:=(a+aa)+(c+cc)
  z:=a+c
  IF ABS(a)>=ABS(c) THEN
    zz:=(((a-z)+c)*scale2+aa)+cc
  ELSE
    zz:=(((c-z)+a)*scale2+cc)+aa
  ENDIF
  IF ABS(z)>0 THEN //underflow protection
    x:=z+zz/scale2
    delx:=x/scale2
    xx:=((((z-x)+zz/scale2)+delx)-delx)*scale2
  ELSE
    x:=z*scale2+zz
    delx:=x/scale2
    xx:=(((x-z)+zz)+delx)-delx
    x:=x/scale2
  ENDIF
ENDPROC longadd2
//
PROC longabs(a,aa,REF x,REF xx) CLOSED
  IF a<0 THEN
    x:=-a; xx:=-aa
  ELSE
    x:=a; xx:=aa
  ENDIF
ENDPROC longabs
```

The routines had two problems that Linnainmaa [2] was not concerned about. The first is that a number such as 1.0 + 1.0e-35 would be represented instead of 1.0 + 0. This gives a false impression of unlimited precision. The second problem was a reduction in the range of the numbers. On the large side, the number may be multiplied by constant:=2^16+1 and cause an overflow. On the small side, the second part of the value is always about 1e-9 of the magnitude of the first part and could underflow. I have fixed these problems in the COMAL routines.

In the representation A=a+aa, I have multiplied aa by 2^32. Now a and aa will have similar magnitudes, and aa carries an implicit scaling. I have also added initial and final scaling to minimize the possibility of overflows and underflows in intermediate calculations.

Exactmul2 gives an exact double precision result of the product of two normal precision numbers. It is basis for the other routines. Longmul and longdiv perform multiplication and division of two double precision numbers. Longadd2 adds two double precision numbers. For subtraction,

more»

simply pass negated values to longadd2. Note that the first part of the number contains the double precision numbers sign and many comparisons can be made by looking only at the first part. If a=0 and A=a+aa then A and aa are both zero. You can take a as the normal precision value of A. For conversion to double precision simply set the second part equal to zero. You could also build other double precision math routines using these basic routines.

Procedure longenviron is the environ procedure converted to double precision. It is an example of how use the double precision routines. If this routine had been for heavy use instead of a straight forward example, the structure would have been changed. As you might expect, it tells us that we are using base 2, 64 digit, proper rounding arithmetic.

```
PROC longenviron(REF beta#,REF t#,REF
round#) // CLOSED Q// wrap line
  // IMPORT longmul,longadd2
  constant:=65537
  maxreal:=1.70141183e+38
  maxtest:=maxreal/2^16
  scale2:=2^32
  round#:=1
  a:=2; aa:=0
  b:=2; bb:=0
  longadd2(a,aa,1,0,x,xx) // X:= A + 1
  longadd2(x,xx,-a,-aa,x,xx) // X:-A
  longadd2(x,xx,-1,0,x,xx) // X:-1
  WHILE x=0 DO //(a+1)-a=1 DO
    longmul(a,aa,2,0,a,aa) // A:= 2*A
    longadd2(x,xx,1,0,x,xx) // X:+1
    longadd2(x,xx,-a,-aa,x,xx) // X:-A
    longadd2(x,xx,-1,0,x,xx) // X:-1
  ENDWHILE
  longadd2(a,aa,b,bb,x,xx) // X:= A + B
  longadd2(x,xx,-a,-aa,x,xx) // X:-A
  WHILE x=0 DO //a+b=a DO
    longmul(b,bb,2,0,b,bb) // B:= 2*B
    longadd2(a,aa,b,bb,x,xx) // X:= A + B
    longadd2(x,xx,-a,-aa,x,xx) // X:-A
  ENDWHILE
  longadd2(a,aa,b,bb,x,xx) // X:= A + B
  longadd2(x,xx,-a,-aa,x,xx) // X:-A
  beta#:=x //beta#:=(a+b)-a
  PRINT "beta=";beta#
  longadd2(a,aa,beta#-1,0,x,xx) //X:=A+(beta#-1)
  longadd2(x,xx,-a,-aa,x,xx) // X:-A
  IF x=0 THEN round#:=0 // see comment below
  //IF a+temp=a THEN round#:=0
  t#:=0
  a:=1; aa:=0
  temp:=beta# // convert beta# to floating point
  REPEAT
    t#:+1
    longmul(a,aa,temp,0,a,aa) // A:+temp
    longadd2(a,aa,1,0,x,xx) // X:= A +1
    longadd2(x,xx,-a,-aa,x,xx) // X:-A
    longadd2(x,xx,-1,0,x,xx) // X:-1
    PRINT t#;a;x
  UNTIL x<>0 //(a+1)-a<>1
ENDPROC longenviron
```

I would still like to see a double precision package. However, these routines are certainly adequate for my purposes. And I hope others can use then also. They are given without proofs. The scaling changes that I have made have not been rigorously tested, and it is possible for some subtle problem to exist. I do expect the C64/COMAL 2.0 to be able to multiply and divide by 2^32 without error.

References:

1. Communications of the ACM, Vol. 15, No. 11, November 1972, pp. 949-951.
2. Linnainmaa, Seppo, *Software for Doubled-Precision Floating-Point Computations*, ACM Transactions on Mathematical Software, Vol. 7, No. 3, September 1981, pp. 272-283.
3. Dekker, T. J. *A Floating-point Technique for Extending the Available Precision*, Number. Math. 18, 1971, pp. 224-242. ■

Files

by David Stidolph

A file is a collection of data held on disk under a single name. This data can represent any type of information you want, but is kept on disk as a sequence of bytes. You can think of a byte as a single character (like the letter A). Bytes on disk are like bytes of memory - you just need a disk drive to read them. Under COMAL there are two types of files - sequential and random. This article deals with sequential files.

Imagine a sequential file as a train with box cars. The engine is the directory entry (filename, type and size, etc.) and the box cars that follow it are the data (one byte per box car). At the end of the train is the caboose which is the **End Of File** marker (in some systems it is a special character, on others it just keeps track of the actual file length).

Writing to a File

In order to work with a file we must first create it. The fastest way to learn about files is to work with them so we will try an example program. The example is to write a program to record test scores and to calculate their average. Because we want to learn about files, the scores will be put in one. The nice feature of keeping the data in a file is that it can be retrieved later.

```
0010 OPEN FILE 2,"scores.dat",WRITE
0020 REPEAT
0030   INPUT "Test score (-1 to quit): ":score
0040   IF score>=0 THEN
0050     PRINT FILE 2: score
0060   ENDIF
0070 UNTIL score=-1
0080 CLOSE FILE 2
```

Now let's look at the program line by line and see how it works.

0010 This is the line that creates the file named scores.dat. The **2** specifies the file number. This is an important number. When you want to read or write information to a file you must use the file number you opened it with. The filename follows the file number (separated by a comma). It can be either a string constant (like in the program) or a string variable. The **WRITE** access type follows the filename (again separated by a comma) and specifies that the file is a sequential type of file and is to be written to only. In addition, the **WRITE** command specifies that the file must be created.

0020 This starts the **REPEAT ... UNTIL** structure. Lines **0030 - 0060** will be repeated until a test score of **-1** is entered.

0030 Prompts the user to enter a test score and assigns the input to the variable **score**.

0040 Tests the variable **score** to make sure it is a positive number (zero or greater). If it is not then line **0050** is skipped.

0050 Prints the variable **score** to the file opened with file number **2**. If the user entered the number **86** the file would contain the following characters:

8	6	cr	Characters
56	54	13	ASCII values

Remember that the number was PRINTed to disk - the file contains the ASCII values that represent each digit of the number (just as if it were printed to the screen). Since the **PRINT FILE** line does not end with a comma (,) or semi-colon (;) a carriage return is also stored in the file.

0060 Ends the **IF .. ENDIF** structure

0070 Tests the variable **score** to see if it is **-1**. If it is not then we jump back to line **0020** to

more»

get another number. If it is then execution continues at line 0080.

0080 Closes the file opened with file number 2. Another option would have been to use the word CLOSE by itself. This would cause all open files to be closed - not just file number 2.

Below is a sample RUN of this program. Comments are in *italics*. The underlined parts are what the user types in:

```
RUN
(the file scores.dat is created and opened)
Test score (-1 to quit): 86
Test score (-1 to quit): 45
Test score (-1 to quit): 92
Test score (-1 to quit): 82
Test score (-1 to quit): 75
Test score (-1 to quit): -1
(the file scores.dat is closed)
```

If you were to examine the file right now you would find the following information:

Value	Character equivalent
56	8
54	6
13	carriage return
52	4
53	5
13	carriage return
57	9
50	2
13	carriage return
56	8
50	2
13	carriage return
55	7
53	5
13	carriage return

Remember that the -1 value that was typed in last was not written to the file. *Note: some implementations of COMAL place a linefeed (ASCII character 10) after the carriage return.*

Reading the File

Now that we have created the file it is time to write a program to read the file and average the scores:

```
0010 OPEN FILE 2,"scores.dat",READ
0020 sum:=0; count:=0
0030 WHILE NOT EOF(2) DO
0040   INPUT FILE 2: score
0050   count:+1
0060   sum:+score
0070   PRINT count;"-->";score
0080 ENDWHILE
0090 CLOSE FILE 2
0100 PRINT "Average is:";sum/count
```

0010 The only difference between this line 0010 and the previous one is that the access type **READ** is used instead of **WRITE**. **READ** is used to open a file already created so that you can read its contents. Reading starts at the beginning of the file.

0020 Since the purpose of this program is to average a series of numbers we need to maintain a count of how many numbers are to be averaged and the total value of all the numbers. Both must start at zero.

0030 We want to read the entire file - yet we do not know how many numbers there are. The EOF function is used to detect the **End Of File**. The 2 inside the parentheses is the file number we are using. The function returns TRUE only after the last data in the file has been read. This loop, then, will continue until the end-of-file has been reached. When end-of-file has been reached, execution jumps to line 0090 (the first line after the ENDWHILE).

0040 The number was printed to disk, so we use INPUT to read it. Remember that there is a carriage return between each number - that way COMAL knows one number from the next.

more»

0050 We have read a number from the file so the number of items (kept in the variable **count**) must be incremented.

0060 The value of the number read from the file is added to the running total (kept in the variable **sum**).

0070 Each test score is printed as it is read in.

0080 This is the ending point of the WHILE .. ENDWHILE loop.

0090 All the numbers in the file have been read so we must now close the file.

0100 With the total number of items and their collective value we can now print their average.

```
RUN
(file is opened)
(file is read - count and sum are updated)
1 --> 86
2 --> 45
3 --> 92
4 --> 82
5 --> 75
Average is: 76
```

Adding to a File

Now that you have created your file, what if you want to add more test scores? COMAL comes to the rescue with the **APPEND** access type. Like **READ** and **WRITE**, it is added after the filename in the **OPEN FILE** line.

```
0010 OPEN FILE 2,"scores.dat",APPEND
0020 REPEAT
0030   INPUT "Test score (-1 to quit): ":score
0040   IF score>=0 THEN
0050     PRINT FILE 2: score
0060   ENDIF
0070 UNTIL score=-1
0080 CLOSE FILE 2
```

The only difference between this program and the original is that **APPEND** is used instead of **WRITE. APPEND** works by opening the already existing file and setting the disk drive so that it will write any new data to the end of the file (adding to it). There is no break between the old data and the new - it stays one file. When you read from the file **EOF** will not return TRUE when you pass the ending point of the original file, only when you reach the actual end of the current file.

Problems with PRINT/INPUT FILE

The method of using **PRINT FILE** and **INPUT FILE** for sending data to a file and retrieving it is useful. The flaw with this method is that you can only store ASCII information (control characters are hard) within the file and that numbers must be separated with commas, carriage returns, or spaces.

```
PRINT FILE 2: students,",",score1,",",score2
```

UniComal Recommends...

A better way, recommended by UniComal (authors of several COMAL systems), is provided by the **READ FILE** and **WRITE FILE** commands. Instead of converting numbers to their ASCII equivalents they are written in their binary form. For example:

```
WRITE FILE 2: students,score1,score2
```

This line is much easier to read and understand. It has many advantages over the first, including being able to read and write any type of data - including strings with control characters. A **READ FILE** command is used to read the data:

```
READ FILE 2: students,score1,score2
```

With READ/WRITE FILE you do not have to worry about data separators or control characters. It is the preferred method. ■

C128 CP/M Sound

by Richard Bain

The C64 COMAL 2.0 cartridge has several sound and music commands built-in (in the **Sound** package). Several sound procedures have been written for COMAL 0.14 (most recently listed in *COMAL Today #15*). Now C128 CP/M COMAL users can also explore the Commodore Sound Interface Device (the SID chip). *Note, the program listed at the end of this article only works in CP/M COMAL running on a C128. It will not work in Commodore COMAL 0.14 or 2.0, nor will it run in CP/M COMAL on non-Commodore CP/M machines.*

The program listed below plays a simple scale. It has been broken down into small procedures plus a function which are designed to be placed in other music programs. The intent of the procedures is to make it as simple as possible for a beginner to get sound out of the SID chip. The procedures have not been cluttered with advanced features of the SID chip such as filtering and ring modulation. Those features are left as an exercise for the reader.

There are a few things a beginner needs to know about sound on the C128 before using the procedures listed below. The SID chip supports three voices numbered from 1-3 (each voice is like a musical instrument in a three piece band). Until you are familiar with making music on your C128, I suggest you limit yourself to the first voice. *Voice 2 is used to ring the bell and make clicks for each keystroke. It should be avoided unless you absolutely need three voices.* To play a note, you must first set its frequency or pitch. You can alter the sound a musical note makes by adjusting its ADSR (attack, decay, sustain, release) wave envelope. Changing its waveform from triangle, to sawtooth, squarewave, or noise will also radically change the sound of a note. Fortunately, a beginner can accept the defaults for these values rather than blindly setting them and hoping for the best. Then, the tone is ready to be turned on and later turned off. Let's look at what some of the procedures below do, and how they are used.

First type in the program and **RUN** it. You should hear a scale being played. If not, turn up the volume and try running the program again. Now you are ready to experiment with the procedures directly. First, try typing this line:

play'note(1,TRUE)

The <u>1</u> indicates voice one and <u>TRUE</u> means to start playing the note. You should now be hearing the computer play middle C. The following command will turn off the note:

play'note(1,FALSE)

As you might have guessed, the <u>1</u> still means voice one and <u>FALSE</u> means to stop playing the note. Now lets try to play C#. First we need to find the frequency of C#, actually we need the number the Commodore uses to translate into C#. The <u>frequency</u> function tells us this number:

PRINT frequency("c#",4)
4547

<u>"C#"</u> is the note you are asking about and <u>4</u> indicates the fourth octave (middle octave) on a piano keyboard. <u>4547</u> is the number returned by the <u>frequency</u> function. It can be used to play the note:

setfrequency(1,4547)
play'note(1,TRUE)

You should now be hearing the computer play a <u>c#</u>. Try using the other procedures to hear how they affect notes. Then put them together into COMAL program to play the music for you. The program below can be used as your guide.

more»

```
use'sound
WHILE NOT EOD DO
  READ note$,octave,duration
  setfrequency(1,frequency(note$,octave))
  play'note(1,TRUE)
  pause(duration)
ENDWHILE
play'note(1,FALSE)
//
PROC use'sound
  // see Commodore 64 Programmer's Ref Guide,
  //     page 461
  DIM frequency'image(3)
  DIM ad'image(3)
  DIM sr'image(3)
  DIM wave'image(3)
  DIM pulse'image(3)
  // The rest of this procedure can
  // be altered or deleted as desired
  volume(15)
  FOR voice:=1 TO 3 DO
    setfrequency(voice,4291)
    pulse(voice,2048)
    waveform(voice,1)
    adsr(voice,8,8,8,8)
  ENDFOR voice
ENDPROC use'sound
//
PROC volume(how'loud)
  IF how'loud<0 OR how'loud>15 THEN
  REPORT 19 // wrap line
  volume'image:=how'loud
ENDPROC volume
//
PROC setfrequency(voice,frequency)
  IF frequency<0 OR frequency>65535
  THEN REPORT 19 // wrap line
  frequency'image(voice):=frequency
ENDPROC setfrequency
//
PROC adsr(voice,a,d,s,r)
  IF a<0 OR a>15 THEN REPORT 19
  IF d<0 OR d>15 THEN REPORT 19
  IF s<0 OR s>15 THEN REPORT 19
  IF r<0 OR r>15 THEN REPORT 19
  ad'image(voice):=16*a+d
  sr'image(voice):=16*s+r
ENDPROC adsr
//
PROC waveform(voice,form)
  // form=1 --> triangle
  // form=2 --> sawtooth
  // form=4 --> square, see PROC pulse
  // form=8 --> noise
  IF form<0 OR form>15 THEN REPORT 19
  wave'image(voice):=form*16
ENDPROC waveform
//
PROC pulse(voice,width)
  IF width<0 OR width>4095 THEN REPORT 19
  pulse'image(voice):=width
ENDPROC pulse
//
PROC play'note(voice,start'stop)
  base:=(voice-1)*7+54272
  FOR x:=base TO base+6 DO OUT x,0
  IF start'stop THEN
    OUT base,frequency'image(voice) MOD 256
    OUT base+1,frequency'image(voice) DIV 256
    OUT base+2,pulse'image(voice) MOD 256
    OUT base+3,pulse'image(voice) DIV 256
    OUT base+5,ad'image(voice)
    OUT base+6,sr'image(voice)
    OUT 54296,volume'image
    OUT base+4,wave'image(voice)+1
  ENDIF
ENDPROC play'note
//
FUNC frequency(note$,octave)
  // see Commodore 64 Programmer's Ref Guide,
  //     page 384
  IF octave<0 OR octave>7 THEN REPORT 19
  CASE LOWER$(note$) OF
  WHEN "c"
    RETURN 34334/(2^(7-octave))
  WHEN "c#"
    RETURN 36376/(2^(7-octave))
  WHEN "d"
    RETURN 38539/(2^(7-octave))
  WHEN "d#"
    RETURN 40830/(2^(7-octave))
  WHEN "e"
```

more»

```
    RETURN 43258/(2^(7-octave))
  WHEN "f"
    RETURN 45830/(2^(7-octave))
  WHEN "f#"
    RETURN 48556/(2^(7-octave))
  WHEN "g"
    RETURN 51443/(2^(7-octave))
  WHEN "g#"
    RETURN 54502/(2^(7-octave))
  WHEN "a"
    RETURN 57743/(2^(7-octave))
  WHEN "a#"
    RETURN 61176/(2^(7-octave))
  WHEN "b"
    RETURN 64814/(2^(7-octave))
  OTHERWISE
    REPORT 19
  ENDCASE
ENDFUNC frequency
//
PROC pause(tenth'of'seconds) CLOSED
  start:=INP(&DC08)
  WHILE tenth'of'seconds>0 DO
    IF INP(&DC08)<>start THEN
      start:=(start+1) MOD 10
      tenth'of'seconds:-1
    ENDIF
  ENDWHILE
ENDPROC pause
//
DATA "c",4,3
DATA "d",4,3
DATA "e",4,3
DATA "f",4,3
DATA "g",4,3
DATA "a",4,3
DATA "b",4,3
DATA "c",5,3
DATA "c",5,3
DATA "b",4,3
DATA "a",4,3
DATA "g",4,3
DATA "f",4,3
DATA "e",4,3
DATA "d",4,3
DATA "c",4,6 ■
```

Shredder

by Luther & Dawn Hux

You are a special investigator and you have been assigned to secure the records in Dawn's office. When you arrive, you find page after page has been through Dawn's shredder. There you are sorting through mounds of shredded paper trying to outsmart Dawn's efforts to keep her records secret. Fortunately, the words are in groups; all you have to do is unscramble each word. Are these the records you're after or just office gossip?

Animated sprites are your reward for attempting to decipher the message by unscrambling the letters. After working on each group of words, you can unscramble the complete message. There are seven scrambled messages paraphrased from the congressional hearings of '87.

This game began as we prepared a classroom lesson on how to animate sprites. The mention of making Dawn's boots walk as possible subject material sounded like a fun way to get our work done. Sprite control in COMAL is much easier than in BASIC so we had the boots up to speed in only a few evenings. The biggest challenge was to decide what boots really look like when walking. Producing a frame-by-frame drawing of the boots in motion made writing the code much easier. An article on how to animate sprites is now in progress, showing how drawings can almost write the code for you.

Once this feature of the hearings was completed, the ideas of turning the shredder into a word scrambler, stuffing paper in the boots and taking advantage of the similarity in names seemed like ideal additions. This program is not intended to make a statement concerning people and events involved in the congressional hearings of '87. It simply uses features and, now famous, phrases to add some spice to the scramble game. Have fun. *Shredder* is on *Today Disk #20* in compiled format. A future issue will explain how the program works. ■

Black Box

by R. L. Brubaker and L. W. Zabel

Blackbox is a strategy game in which the player attempts to find a series of markers hidden by the computer. There have been other computer versions of this game, but ours is the first, that we know about, written in COMAL. Complete and detailed on-screen instructions are included. When the player requests instructions, the system plays a demonstration game so that the player can become familiar with the moves and the results of each move. When playing a game against the computer, the maximum possible score is 20. If a player's score is half the maximum, he is doing well.

[*Editor's note: Blackbox works unchanged in CP/M COMAL and UniComal IBM PC COMAL. It does not work on the C64 COMAL cartridge because it uses 80 column screen output.*]

Instructions

The game is played on an 8 x 8 grid. Somewhere in the grid I will hide 4 secret markers. Your mission is to determine where they are located. There are 32 openings around the outside of the grid (8 on each side), which I have numbered corresponding to one of the openings. Imagine that you are aiming a probing ray into the grid. There are several results you may observe:

1. The ray may hit one of the secret markers and be totally absorbed so it never emerges.
2. The ray may be reflected back at you. Don't worry, it doesn't hurt much.
3. The ray may pass through the grid and will then emerge from the opening on the opposite side of the grid.
4. The ray may be deflected and will then emerge at some other opening which you may not have expected.

I will tell you what has happened by marking the starting opening with **HIT** for the first case, **REF** for the second case and both openings with an identifying letter in the latter cases. The behavior of these rays will provide clues about the locations of the markers. But first you will have to understand some things about the behavior of the probing rays. Normally the rays follow straight horizontal or vertical paths. However, they cannot pass a secret marker on an adjacent grid path. Thus, when their path would violate this rule, they are forced to turn. They may turn either 90 degrees or when that is not possible, they will turn 180 degrees. The latter leads to REFlections. If a secret marker lies directly in their path they will HIT it, even if there is another marker directly along side of it that would otherwise cause it to turn. A ray that tries to enter the grid may also be reflected by a marker that is on either side of the opening it tries to enter.

To help you visualize where the secret markers are I will provide you with a means of placing your own markers. You will have to tell me when you want to do this by typing the letter «m» instead of a number. You may place a marker at any square that you desire by entering the row and column numbers whose coordinates cross at the square you have selected. If you want to place a marker type an «a» (for Add). You can Erase a marker with the letter «e». You will automatically go back to the number entry mode after placing a marker. To place another marker, just press «m» again and repeat the process. Place markers wherever you think that there are hidden markers. When you think that your markers are all in the correct locations enter the letter «s» and I will check your results and tell you your score. The maximum possible score is 20, five times the number of markers.

After every entry, be sure to press «*return*».

more»

```
DIM l(10,10), p(10,10)
DIM xl(4), yl(4)
DIM marker(4)
LOOP
  start
  ask'demo
  draw'diagram
  hide'markers
  screen'inst
  play'game
ENDLOOP
//
PROC start
  ix:=0; iy:=0; x:=0; y:=0
  FOR i:=1 TO 4 DO
    marker(i):=0
    xl(i):=0; yl(i):=0
  ENDFOR i
  FOR i:=1 TO 10 DO
    FOR j:=1 TO 10 DO
      l(i,j):=0; p(i,j):=0
    ENDFOR j
  ENDFOR i
  character:=65 //counter to get next ascii char
  trys:=0 // counter used for solution
ENDPROC start
//
PROC ask'demo
  demo'mode:=FALSE
  PAGE
  INPUT "Do you want to see a demo? ": reply$
  IF reply$ IN "YESyes" THEN demo'mode
  :=TRUE // wrap line
ENDPROC ask'demo
//
PROC draw'diagram
  PAGE
  FOR i#:=3 TO 19 STEP 2 DO
    FOR j#:=38 TO 76 DO
      PRINT AT i#,j#: "-"
    ENDFOR j#
  ENDFOR i#
  FOR i#:=41 TO 73 STEP 4 DO
    FOR j#:=2 TO 20 DO
      PRINT AT j#,i#: CHR$(124)
    ENDFOR j#
  ENDFOR i#
  PRINT AT 2,42: "24 !23 !22 !21 !20 !19 !18
  !17 !" // wrap line
  PRINT AT 4,39: "25"
  PRINT AT 4,74: "16"
  PRINT AT 6,39: "26"
  PRINT AT 6,74: "15"
  PRINT AT 8,39: "27"
  PRINT AT 8,74: "14"
  PRINT AT 10,39: "28"
  PRINT AT 10,74: "13"
  PRINT AT 12,39: "29"
  PRINT AT 12,74: "12"
  PRINT AT 14,39: "30"
  PRINT AT 14,74: "11"
  PRINT AT 16,39: "31"
  PRINT AT 16,74: "10"
  PRINT AT 18,39: "32"
  PRINT AT 18,74: "9"
  PRINT AT 20,42: " 1 ! 2 ! 3 ! 4 ! 5 ! 6 ! 7
  ! 8 !" // wrap line
  PRINT AT 1,10: "BLACK BOX"
  PRINT AT 2,10: "========="
ENDPROC draw'diagram
//
PROC hide'markers
  RANDOMIZE
  n:=4
  mark$:="four"
  FOR i:=1 TO n DO
    REPEAT
      marker(i):=RND(1,64)
      match:=FALSE
      FOR j:=1 TO i-1 DO
        IF marker(i)=marker(j) THEN match
        :=TRUE // wrap line
      ENDFOR j
    UNTIL NOT match
  ENDFOR i
  IF demo'mode THEN
    PRINT AT 3,6: "Find ";mark$;" markers.
    DEMO MODE" // wrap line
  ELSE
    PRINT AT 3,6: "Find ";mark$;" markers.
    Good Luck!" // wrap line
  ENDIF
```

more»

```
  FOR i:=1 TO n DO
    xl(i):=1+(marker(i)-1) MOD 8
    yl(i):=1+(marker(i)-1) DIV 8
  ENDFOR i
  IF demo'mode THEN
    FOR i:=1 TO n DO
      PRINT AT (2+2*yl(i)),(4*xl(i)+39): "X"
    ENDFOR i
  ENDIF
  FOR i:=1 TO n DO
    l(yl(i)+1,xl(i)+1):=1
  ENDFOR i
ENDPROC hide'markers
//
PROC screen'inst
  PRINT AT 9,6: "MARKER MODE: Enter
  markers" // wrap line
  PRINT AT 10,6: "by entering an M<RET>
  then" // wrap line
  PRINT AT 11,6: "enter the row<RET> and
  then the" // wrap line
  PRINT AT 12,6: "column<RET> that cross at
  the" // wrap line
  PRINT AT 13,6: "point where you want a
  marker." // wrap line
  PRINT AT 14,6: "Add or Erase markers using:"
  PRINT AT 15,6: "A<RET> for Add  E<RET>
  for Erase" // wrap line
  PRINT AT 16,6: "You will then be returned"
  PRINT AT 17,6: "to the NUMBER ENTRY
  MODE." // wrap line
  PRINT AT 19,6: "SCORE MODE: For the
  Solution" // wrap line
  PRINT AT 20,6: "and Score enter an S<RET>."
ENDPROC screen'inst
//
PROC play'game
  LOOP
    PRINT AT 5,6: SPC$(32)
    PRINT AT 6,6: SPC$(32)
    PRINT AT 5,6: "Enter a number (1-32)"
    INPUT AT 5,27,2: reply$
    CASE reply$ OF
    WHEN "m","M"
      IF demo'mode THEN
        PRINT AT 5,6: "Manual setting of
        markers" // wrap line
        PRINT AT 6,6: "in Demo Mode not
        allowed" wrap line
        pause(1500)
      ELSE
        set'markers
      ENDIF
    WHEN "S","s"
      end'routine
      EXIT
    OTHERWISE
      TRAP
        t:=VAL(reply$)
        IF t<1 OR t>32 THEN
          PRINT AT 6,6: "Wrong number, try
          again" // wrap line
          pause(1500)
        ELSE
          trace'init
          trace'paths
        ENDIF
      HANDLER
        PRINT AT 6,6: "Only numbers or m or
        s allowed." // wrap line
        pause(1500)
      ENDTRAP
    ENDCASE
  ENDLOOP
ENDPROC play'game
//
PROC trace'init
  trys:+1
  IF t>0 AND t<9 THEN
    ix:=0; iy:=-1; x:=t; y:=9
  ELIF t>8 AND t<17 THEN
    ix:=-1; iy:=0; x:=9; y:=17-t
  ELIF t>16 AND t<25 THEN
    ix:=0; iy:=1; x:=25-t; y:=0
  ELSE // t>24 AND t<33
    ix:=1; iy:=0; x:=0; y:=t-24
  ENDIF
  start'x:=x; start'y:=y
ENDPROC trace'init
//
PROC trace'paths
  REPEAT
```

more»

```
    IF ix=0 THEN
      vertical
    ELSE
      horizontal
    ENDIF
    continue:=FALSE
    IF h=-1 THEN // Hit
      PRINT AT 7,6: "Its a HIT!!"
      y1:=2+2*start'y; x1:=38+4*start'x
      PRINT AT y1,x1: "HIT"
      h:=0
    ELIF x=0 OR x=9 OR y=0 OR y=9 THEN
    // ray emerged
      IF x=start'x AND y=start'y THEN
        PRINT AT 7,6: "A Reflection"
        y1:=2+2*y; x1:=38+4*x
        PRINT AT y1,x1: "REF"
      ELSE
        y1:=2+2*y; x1:=38+4*x
        PRINT AT y1,x1:" "+CHR$(character)+" "
        y1:=2+2*start'y; x1:=38+4*start'x
        PRINT AT y1,x1:" "+CHR$(character)+" "
        character:+1
        PRINT AT 7,6: SPC$(30)
      ENDIF
    ELSE
      IF demo'mode=TRUE THEN PRINT AT
      (2+2*y),(39+4*x): "*" // wrap line
      continue:=TRUE
    ENDIF
  UNTIL NOT continue
ENDPROC trace'paths
//
PROC vertical
  h:=0
  IF l(y+iy+1,x+1)=1 THEN
    h:=-1
  ELIF l(y+iy+1,x+2)=-1 AND l(y+iy+1,x)=1 THEN
    iy:=-iy
  ELIF l(y+iy+1,x+2)=1 THEN
    ix:=-1; iy:=0
  ELIF l(y+iy+1,x)=1 THEN
    ix:=1; iy:=0
  ELSE
    y:=y+iy
  ENDIF
ENDPROC vertical
//
PROC horizontal
  h:=0
  IF l(y+1,x+ix+1)=1 THEN
    h:=-1
  ELIF l(y+2,x+ix+1)=-1 AND l(y,x+ix+1)=1 THEN
    ix:=-ix
  ELIF l(y+2,x+ix+1)=1 THEN
    ix:=0; iy:=-1
  ELIF l(y,x+ix+1)=1 THEN
    ix:=0; iy:=1
  ELSE
    x:=x+ix
  ENDIF
ENDPROC horizontal
//
PROC set'markers
  REPEAT
    valid'move:=TRUE
    PRINT AT 5,6: SPC$(32)
    PRINT AT 6,6: SPC$(32)
    PRINT AT 7,6: SPC$(32)
    PRINT AT 5,6: "Enter row number"
    INPUT AT 5,25,2: r
    PRINT AT 6,6: "Enter column number"
    INPUT AT 6,27,2: c
    IF r>24 AND r<33 THEN
      r:=r-24
    ELIF r>8 AND r<17 THEN
      r:=17-r
    ELSE
      valid'move:=FALSE
    ENDIF
    IF c>0 AND c<9 THEN
      NULL
    ELIF c>15 AND c<25 THEN
      c:=25-c
    ELSE
      valid'move:=FALSE
    ENDIF
    IF NOT valid'move THEN
      PRINT AT 7,6: "Invalid number, try again"
      pause(250)
    ENDIF
  UNTIL valid'move
```

more»

```
  REPEAT
    PRINT AT 5,6: SPC$(32)
    PRINT AT 6,6: SPC$(32)
    PRINT AT 5,6: "Add or Erase marker?
    (A/E)" // wrap line
    INPUT AT 5,33,1: reply$
    IF reply$ IN "Aa" THEN
      p(r,c):=1
      PRINT AT (2+2*r),(39+4*c): "#"
    ELIF reply$ IN "Ee" THEN
      PRINT AT (2+2*r),(39+4*c): " "
      p(r,c):=0
    ELSE
      PRINT AT 6,6: "Wrong response, try
      again" // wrap line
      pause(250)
    ENDIF
  UNTIL reply$ IN "AaEe"
ENDPROC set'markers
//
PROC end'routine
  score:=0
  FOR i:=1 TO n DO
    PRINT AT (2+2*yl(i)),(38+4*xl(i)): "x"
    IF p(yl(i),xl(i))=1 THEN score:=score+1
  ENDFOR i
  PRINT AT 22,6: trys;" numbers used. ";" Score
  is ",5*score-trys // wrap line
  INPUT AT 23,2,1: "Do you want to play
  again? (Y/N) ": reply$ // wrap line
  IF NOT reply$ IN "Yy" THEN
    END "I hope that you enjoyed the game.
    Bye" // wrap line
  ENDIF
ENDPROC end'routine
//
PROC pause(number)
  FOR delaying:=1 TO number DO NULL
ENDPROC pause
//
PROC new'page
  PRINT
  PRINT "Press <RETURN> to continue"
  WHILE KEY$="" DO NULL
  PAGE
ENDPROC new'page ■
```

Mod Tutorial

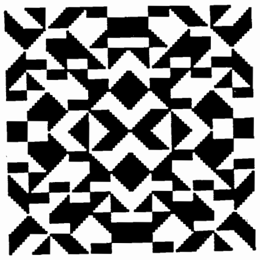
by Bill Inhelder

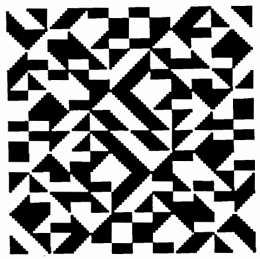
Mod'tutorial on *Today Disk #20* is a tutorial and demonstration program illustrating how a modulus N multiplication table can be used to generate graphic designs. The basic design patterns created by various modulus numbers may then be used to produce enlarged patterns using the principles of horizontal, vertical and central symmetry.

By varying the assignment of different graphic characters other interesting patterns can be produced.

The hardcopy illustrations accompanying this article were produced by using the procedure **copyscreen** in *Comal Today #13* to copy the text screen to the high resolution screen. Your favorite high resolution screen dump to the printer completes the task.

For an unusual pattern, press the shift and Commodore buttons to change from upper case/graphics mode to lower/upper case mode when the mod 5 graphs are displayed.


horizontal and vertical symmetry


central symmetry ■

Disk Directory Sleeves

Today Disk #18 - Front

```
                                                     86 Files          1 Blocks Free:
bootslow          ~ programs ~      random'walk       ~ please type~ ~  font.nu'deco
----------------  ----------------  tiny'comal        ----------------  set.gumby.a
c64 comal 0.14    1520dir'print     ----------------  ~ new         ~  ----------------
comalerrors       3circle/octs      ~these programs~  ~ enter <name> ~  ~for more info ~
ml.sizzle         3rd'dimension     ~chain from one~  ~ run         ~   ~ send sase to ~
hi                amortization      ~ to the next ~   ----------------  ----------------
menu              blackjack'advice  ----------------  ~note~ you must~  ~ comal users ~
names.dat         convert'ps/pm     the'easter'star   ~use quote mark~  ~ group, usa ~
----------------  decimal'rep'orig  trisqoct          ~around name  ~   ~ 6041 monona ~
~ text files ~    decimal'rep'2     special'ball      ----------------  ~ madison, wi ~
----------------  demo/load'font    build'a'sun       binomial.lst      ~   53716    ~
comal article     geometry'lesson   ----------------  grammar.lst       ----------------
info.txt          haiku             ~ listed comal ~  hamurabi.lst      ~(608)222-4432 ~
keywords.txt      knights'tour      ~ programs for ~  ----------------  ----------------
graphics.txt      new'typing'tutor  ~ all versions ~  ~ font files ~
sprites.txt       paint'circle      ----------------  ----------------
----------------  random'lines      ~ to execute ~    font.astronomy
~ comal 0.14 ~    random'turtle     ~these programs~  font.gumby
```

Today Disk #18 - Back

```
                                                     101 Files         2 Blocks Free:
hi                epicycloids       proc.ramall       ----------------  ~ listed progs ~
----------------  haiku             ----------------  bat.memory        ~ on the 0.14 ~
~ comal 2.0  ~    knights'tour      ~  external  ~    crypt2.dat        ~ side of this ~
~  programs  ~    mandelbrot        ~ procedures ~    ssdemo1.dat       ~     disk     ~
----------------  mandelbrot'small  ----------------  ssdemo2.dat       ----------------
3rd'dimension     new'typing'tutor  ssdemo1.ext       upszones.dat      ----------------
big'print         printshop'loader  ssdemo2.ext       ----------------  ~for more info ~
blackjack'advice  puzzle            ----------------  ~ print shop ~    ~ send sase to ~
chip.circle80     relinker          ~ source code ~   ~ icon files ~    ----------------
comal-flex(joy)   reversi           ----------------  ----------------  ~ comal users ~
crypto'solver     spread'sheet      src.memory        icon.donald'duck  ~ group, usa ~
cryptogram        twin'drivecopy    ----------------  icon.garf'head    ~ 6041 monona ~
decimal'rep'orig  ups'chart         ~  packages  ~    ----------------  ~ madison, wi ~
decimal'rep'2     ups'zone          ----------------  ~ flexi fonts ~   ~   53716    ~
decimal'rep'fin   ----------------  pkg.clock         ----------------  ----------------
demo/clock        ~ procedure  ~    pkg.memory        /avante garde     ~(608)222-4432 ~
demo/mouse        ----------------  pkg.mouse         /times            ----------------
demo/procs        proc.copyscreen   pkg.spiritdump    ^avante garde
demo/ups          proc.flexi'load   pkg.ps'convert    ^times
edit'src          proc.large        ----------------  ----------------
encrypt           proc.procs        ~ data files ~    ~ there are 3 ~
```

Today Disk #19 - Front

```
                                                     84 Files          1 Blocks Free:
bootslow          ~ comal 0.14 ~    ----------------  f.dat             w.dat
fastboot          ~  programs  ~    alphabet soup     g.dat             x.dat
c64 comal 0.14    ----------------  header            h.dat             y.dat
comalerrors       1520stereo'plane  header+c          i.dat             z.dat
ml.sizzle         3d'probability    header+co         j.dat             ----------------
hi                check'random      header+comal      k.dat             ~for more info ~
menu              coloring'book     tut1              l.dat             ~ send sase to ~
names.dat         demo/rotate'3d    tut2              m.dat             ----------------
----------------  get'rid'of'boxes  tut3              n.dat             ~ comal users ~
~ text files ~    hi-lo-1           ----------------  o.dat             ~ group, usa ~
----------------  hi-lo-2           ~ data files ~    p.dat             ~ 6041 monona ~
comal article     mandelbrot2       ----------------  q.dat             ~ madison, wi ~
info.txt          rotate'3d'plane   a.dat             r.dat             ~   53716    ~
keywords.txt      ----------------  b.dat             s.dat             ----------------
graphics.txt      ~these prorams ~  c.dat             t.dat             ~ 608-222-4432 ~
sprites.txt       ~ chain each ~    d.dat             u.dat             ----------------
----------------  ~    other   ~    e.dat             v.dat
```

Background:

Beginning with COMAL Today #11, we have been printing the directories of our disks in a special format.

Each disk directory listing is printed in 5 columns inside an area that will fit within the front of your disk sleeve.

Just photocopy the page, and cut out the directories. Then use rubber cement, glue stick or other adhesive and attach the directory listing onto the front of your paper/tyvek disk sleeve.

We have provided disk sleeve ready directories for all of our disks now. The name of the disk is in larger type for easier identification. Also, the total number of files and the number of free blocks is provided on the top line.

Now, using these directory pages, you can customize your COMAL disks easily. For each COMAL disk that you have, just find the issue of COMAL Today that provides its directory. All the early disks are listed in issue #11. Then as the disks are released they are listed in the next possible newsletter (approximately).

more»

POWER SUPPLY:

FOR SALE: CPS-30 C64 Power Supply, repairable, short circuit protected, used for 2 months: $30 Sister Anne Stremlau, 329 Mansion St, Mauston, WI 53948

Math Program:

Derivative is a program designed for use in High School Pre-Calculus courses which discusses the development of the limit definition of the first derivative of a polynomial function. It is assumed that the student who uses this program has been well grounded in Algebra and Geometry, and has worked with slope, polynomial functions, tangent lines, secant lines, and limits. The program is intended to aid the student in understanding how the first derivative of a polynomial function is developed from taking the limit of the slopes of secant lines approaching a given point. The student is then able to plot any polynomial function, select any point, and view the secant lines approaching that point. The power formula is also developed as an alternate and easier method for finding the first derivative A Teacher Guide and a Student Guide are provided. For more information contact:

Larry Winckles
Toledo Christian Schools
432 S. Harefoote
Holland, OH 43528
(419) 865-6926

Today Disk #19 - Back

			62 Files	2 Blocks Free:
hi	demo/rotate'3d	----------------	lite.hello	- send sase to -
----------------	epidemic	ext.ghosts	hrg.introghosts	----------------
- comal 2.0 -	get'rid'of'boxes	----------------	----------------	- comal users -
- programs -	ghosts	- listed files -	- flexi-fonts -	- group, usa -
----------------	graph'error'fix	----------------	----------------	- 6041 monona -
1520stereo'plane	hi-lo-1	benchmark.lst	- do not load -	- madison, wi -
1541'aligner	hi-lo-2	func.value	----------------	- 53716 -
3d'surfaces	lite'byte	proc.mandelbrot	/avante garde	----------------
benchmark	mandelbrot2	----------------	/times	- 608-222-4432 -
cave'warrior	recover'files	- data files -	^avante garde	----------------
chip.mandelbrot	smarter'reader	----------------	^times	
comal-flex(joy)	----------------	dat.ghost	----------------	
cursor'fun	~external proc ~	lite.grid	~for more info ~	

Power Driver

			119 Files	1 Blocks Free:
bootslow	free'form'db	sound'effects	- procedures -	- comalites -
fastboot	hi	----------------	----------------	- united -
ml.sizzle	microscope'quiz	- compiled -	buffer.proc	- 1670 simpson -
----------------	program'outliner	- program -	circle.proc	- #102 -
- power driver -	rotate'3d'plane	----------------	clear'keys.proc	- madison, wi -
----------------	----------------	- load from -	create.proc	- 53713 -
power driver	~data files for~	- basic -	drive8.proc	----------------
----------------	~programs above~	----------------	drive9.proc	----------------
- power driver -	----------------	coloring.system	joystick.proc	~this disk can ~
- compiler -	comal article	----------------	koala.proc	~not be put in ~
----------------	dat.ffdb	- power driver -	loadshape.proc	- user group -
compiler	info.txt	- functions -	read'dir.proc	- libraries or -
----------------	microscope.dat	----------------	saveshape.proc	- uploaded on -
- comal 0.14 -	names.dat	blocks'free.func	showtable.proc	- bbs system -
- programs -	notes.txt	countsp.func	textcolors.proc	----------------
----------------	----------------	decimal.func	type.proc	- you may make -
- you may load -	- power driver -	file'exists.func	----------------	- backups of -
- and run them -	- programs -	frac.func	- power driver -	- this disk or -
~from comal .14~	----------------	freefile.func	- and -	~any file on it~
- or -	- you can run -	round.func	- power driver -	~only for your ~
- power driver -	~them only from~	trapped.func	- compiler -	- own personal -
----------------	- power driver -	trunc.func	----------------	- use -
bill'paint	----------------	----------------	~copyright 1987~	----------------
coloring'book	read'dir	- power driver -	----------------	

0.14 Functions/Procedures

			126 Files	4 Blocks Free:
bootslow	graphics'on.func	drive9.proc	plottext.proc	- procedures -
fastboot	peek'hires.func	dump'1525.proc	print'at.proc	----------------
c64 comal 0.14	pi.func	dump1520.proc	quicksort.proc	1520driver.lst
comalerrors	round.func	epson'cardg.proc	read'errors.proc	1571.lst
ml.sizzle	sigdig.func	exchange.proc	repeat'key.proc	base'convert.lst
hi	spritecolor.func	expand'ram.proc	repeatkeys.proc	bit'funtions.lst
menu	trunc.func	fillkeys.proc	restore'lbl.proc	fx-80'cmds.lst
names.dat	val.func	front'side.proc	restore'scn.proc	graph'system.lst
window-rs232.obj	----------------	inkey.proc	save'obj.proc	ml'procs.lst
----------------	- procedures -	input'at.proc	save'screen.proc	read'block.lst
- programs -	----------------	joystick.proc	saveshape.proc	sound'system.lst
----------------	back'side.proc	koala.proc	select'sort.proc	----------------
listerine	bubblesort.proc	lightpen.proc	set'text.proc	- articles -
merge'procs	bubblesort2.proc	line'length.proc	settime.proc	----------------
----------------	bubblesort3.proc	load'comp.proc	shift'wait.proc	book reviews.txt
- functions -	buffer.proc	load'errors.proc	showtable.proc	comal article
----------------	call.proc	load'font.proc	shuffle'in.proc	comal today.txt
curcol.func	circle.proc	load'obj.proc	singlesided.proc	graphics.txt
currow.func	clear'keys.proc	load'sizzle.proc	str.proc	info.txt
decimal.func	clearkeys.proc	loadfont.proc	textcolors.proc	keywords.txt
file'exists.func	compact'14.proc	loadshape.proc	turbo.proc	procs&funcs.txt
frac.func	create.proc	mount.proc	use'sound.proc	sprites.txt
free.func	cursor.proc	paddle.proc	----------------	
freefile.func	dir.proc	page.proc	- listed files -	
getbackgrnd.func	doublesided.proc	payment.proc	----------------	
gettime.func	drive8.proc	plot'text.proc	- groups of -	

Superchip Source Code

```
                                                      74 Files          148 Blocks Free:
hi                src.c128-part2    ~ program ~       ~ text files ~    ~group library ~
----------------  src.c128-part3    ----------------  ----------------  ~or bbs system ~
~comal programs~  src.c128-part4    ~ use prog'ram ~  txt.assembling    ----------------
----------------  src.c128-keypad   ~ to turn into ~  txt.c128graphics  ~for more info ~
prog'ram          src.system2       ~binary package~  ----------------  ~ send sase to ~
change'autostart  src.math          ----------------  ~this disk and ~  ----------------
----------------  src.strings       ~ use ~           ~ the source ~    ~ comal users ~
~ superchip ~     src.keyboard      ~ change'auto ~   ~code on it is ~  ~ group, usa ~
~ source code ~   src.colors        ~to change load~  ~ copyrighted ~   ~ 6041 monona ~
----------------  src.files         ~address $5000 ~  ~ 1986 ~          ~ madison, wi ~
src.chip-nolist   src.rabbit-1      ----------------  ----------------  ~ 53716 ~
src.chip-list     src.rabbit-2      autostart         ~you cannot put~  ----------------
sym.comal         src.rabbit-3      lst.autostart     ~ this disk or ~  ~(608)222-4432 ~
src.header        ----------------  bin.autostart     ~ the files on ~  ----------------
src.c128-part1    ~ autostart ~     ----------------  ~it in any user~
```

Package Library Volume 2

```
                                                      88 Files          141 Blocks Free:
hi                ~ disassembler ~  ~convert binary~  pkg.gemini        ~ allowed to ~
----------------  ----------------  ~ file to hex ~   pkg.irq           ~ make copies ~
~ listed progs ~  disassem20        ~file for link ~  pkg.kol80         ~ for your own ~
----------------  ----------------  ----------------  pkg.matrix        ~ personal use ~
lst.delink        ~ relink comal ~  bin'to'pkg        pkg.memory        ----------------
lst.list'package  ~ program with ~  ----------------  pkg.meta          ~for more info ~
----------------  ~fonts, sprites~  ~ packages ~      pkg.meta'rommed   ~ send sase to ~
~ procedure to ~  ~ and packages ~  ----------------  pkg.mlw'sorts     ----------------
~ package prog ~  ----------------  pkg.basic         pkg.mouse         ~ comal users ~
----------------  relinker          pkg.bits          pkg.pllprt        ~ group, usa ~
package'maker     ----------------  pkg.blas          pkg.ps'convert    ~ 6041 monona ~
lib.labels        ~ program to ~    pkg.cleanup       pkg.screenhelp    ~ madison, wi ~
----------------  ~ linkable or ~   pkg.clock         pkg.spiritdump    ~ 53716 ~
~batch file to ~  ~ romable ~       pkg.clock2        pkg.splitscleft   ----------------
~ package prog ~  ~ package ~       pkg.code'doctor   pkg.text          ~ 608-222-4432 ~
----------------  ----------------  pkg.dualscreen    pkg.windows       ----------------
batch'to'package  prog'ram          pkg.dvorak        ----------------
----------------  ----------------  pkg.environment   ~ you are only ~
```

Package Library 2 - Back

```
                                                      50 Files          6 Blocks Free:
----------------  ----------------  src.dvorak        src.printer       ~ send sase to ~
~ comal symbol ~  ~ source code ~   src.environment   src.ps'convert    ----------------
~ file ~          ----------------  src.gemini        src.screenhelp    ~ comal users ~
----------------  src.basic         src.irq1.00       src.spiritdump    ~ group, usa ~
symbs             src.bits          src.irq2.01       src.splitscleft   ~ 6041 monona ~
----------------  src.blas          src.kol80         src.text          ~ madison, wi ~
~data table for~  src.clock         src.matrix        src.version       ~ 53716 ~
~ pkg.kol80 ~     src.clock2        src.memory        src.windows       ----------------
----------------  src.code'doc      src.meta          ----------------  ~ 608-222-4432 ~
lib.table80       src.customdump    src.mouse         ~for more info ~  ----------------
```

COMAL Collage Disk

```
                                                      109 Files         310 Blocks Free:
----------------  ~coloring book ~  morse             ----------------  mental
~ trip of the ~   ----------------  scenemus          sprtmel           mental2
~ troubled ~      color             ----------------  imag.melody       mental3
~ turtle ~        paintsize         ~ useful ~        readmly           rlx'talk
----------------  pie'eyed          ~ utilities ~     imag.spooky       ----------------
new'turtle        frame             ----------------  imag.bigdaddy     ~copyright 1987~
square'proc       flower'stamp      proc.pause        imag.rhianon      ~by comal users~
square2'proc      xmas'tree         proc.hues         imag.target       ~group, u.s.a.,~
polygon           composite'print   bag'em            ----------------  ~ limited ~
indian'attack     motion            infile.dat        ~ integrated ~    ----------------
----------------  tumble'sticks     multitab          ~ activities ~    ~for more info ~
~ the joy of ~    line'pattern      plotit            ----------------  ~ send sase to ~
~ joysticks ~     night'grass       sumit             combination       ----------------
----------------  night'waves       avgit             hrg.night'grass   ~ comal users ~
joyhsekp          ----------------  frame'it          ----------------  ~ group, usa ~
joystick          ~ sounds and ~    viewport          ~ advanced ~      ~ 6041 monona ~
joyplot           ~ songs ~         window            ~ features ~      ~ madison, wi ~
pic'action        ----------------  viewwind          ----------------  ~ 53716 ~
pic'game          noodling          univarea          speak             ----------------
joychs2           octave            ----------------  speak2            ~(608)222-4432 ~
----------------  autosong          ~ splendid ~      talk              ----------------
~ a comal ~       joysound          ~ sprites ~       talk2
```

The JADE Disk:

QLink Message
SUBJ: The 'JADE' disk!
FROM: XAX 01
DATE: 11/30/87
S#: 819349

This disk contains valuable programs and procedures to aid in program development and system maintenance. Perfect for the COMAL enthusiast. Send $7.20 check or money order to:

Christopher Laprise
5700 Carbon Canyon Road
#118
Brea, CA 92621

or call:

(714) 524-0343

State your name and address clearly. Request the 'JADE' disk!

All code is conscientiously written for speed and clarity and contains no protection. All code requires the presence of Super Chip or its packages to operate as it is such a standard. Rabbit routines are employed wherever possible! (i.e, these programs are fast).

Includes:

MASTER DIRECTORY disk catalog management program with:

Global SEARCH with optional DELETE, catalog entry from KEYBOARD, or DIRECTORY. Sorting and category maintenance. Completely compatible with Disk File 4.1 Files (by ramsoft). Conversion routines for DCMR files! 800 entries/file, 10 categories/disk capacity. »

Utility for hardcopies of your categories @ 3 cols/page. Includes example files!

RABBIT FILE COPIER
Can copy multiple files per pass. First "fastfile" copier for the COMAL environment.

IMPOSSIBLE FILE RECOVERY PROGRAM (revised): Recovers even from disks whose directories have been completely overwritten.

FILE / DISK ENCRYPTION SYSTEM: Key-based RABBIT encoder/decoder for file-by-file or sector-by-sector encryption!

DELUXE SEQ FILE EDITOR
Edit up to 80 column files (78 columns visible in scroll mode) with up-down-left-right cursor control text scrolling, file optimizing, and much more.

HI-RES SLIDE PUZZLE
Reconstruct the images in the least amount of moves possible! 5x5 grid. Lets you go back and see the puzzle "solved" during game. Includes facilities to make your own puzzles out of your COMAL "HRG." files! Includes examples!

RABBIT DIRECTORY PROCEDURES
PROC dir'
PROC get'files
Closed callable procedures for programs that can automatically manipulate files! Examples included.

Compact Picture Disk #1

			80 Files	7 Blocks Free:
----------------	compact pix	keyboard.crg	piechart.crg	- libraries or -
- comal 2.0 -	b & o.crg	liberty.crg	pirate&poly.crg	- uploaded to -
- program -	bbs.crg	limosine.crg	sesame.st.crg	- any bbs -
----------------	blackcomal.crg	linefig1.crg	sgt.major.crg	----------------
crg2	business car.crg	linefig2.crg	sincos1.crg	----------------
----------------	butterfly.crg	linefig3.crg	tunnel.crg	-for more info -
- comal 0.14 -	cabbage patc.crg	linefig4.crg	watch.crg	- send sase to -
- programs -	candle1.crg	lone pine.crg	----------------	----------------
----------------	caveman.crg	loon.crg	- you may make -	- comal users -
crg14.filewriter	city.crg	map.crg	- copies for -	- group, usa -
crg14.compactor	cos surface.crg	middle earth.crg	-yourself only -	- 6041 monona -
crg14.viewer	dip.crg	mirror1.crg	----------------	- madison, wi -
----------------	dollar.crg	music.crg	- the files on -	- 53716 -
- compacted -	goofy.crg	needlepoint.crg	- this disk -	----------------
- pictures -	goofy1.crg	objects.crg	-cannot be put -	- 608-222-4432 -
----------------	heart.crg	petroom.crg	-in user group -	----------------

Compact Picture Disk #2

			66 Files	11 Blocks Free:
----------------	fremen chief.crg	paul/student.crg	-this disk may -	- send sase to -
- compacted -	geom1.crg	paul/trainin.crg	- copied for -	----------------
- pictures -	guard.crg	rabban.crg	- your own use -	- comal users -
----------------	guards.crg	seahorse2.crg	- only -	- group, usa -
compact pix	guild heighl.crg	seed.crg	----------------	- 6041 monona -
baron & dr.crg	guild nav.crg	shel's circs.crg	- this disk -	- madison, wi -
box of truth.crg	han solo.crg	shelly2.crg	-cannot be put -	- 53716 -
br harkonnen.crg	indian.maide.crg	spice miners.crg	-in user group -	----------------
captive luke.crg	jabba & lea.crg	stage 3 nav.crg	- libraries or -	- 608-222-4432 -
duke & fam.crg	jabba court.crg	talk/boush.crg	- uploaded to -	----------------
dukes castle.crg	landwalker.crg	train1.crg	- any bbs -	
emporor.crg	larry payne.crg	vader/combat.crg	----------------	
ewok warrior.crg	paul & feyd.crg	----------------	----------------	
ewoks.crg	paul & mapes.crg	- the files on -	-for more info -	

Guitar Tutorial - System

			34 Files	104 Blocks Free:
bootslow	- comal 0.14 -	how'chords	tuning	- 6041 monona -
fastboot	- programs -	how'tab	----------------	- madison, wi -
c64 comal 0.14	----------------	how'to'hold	-for more info -	- 53716 -
comalerrors	instructions	kcn	- send sase to -	----------------
ml.sizzle	chords	main'menu	----------------	- 608-222-4432 -
hi	chromatic'scale	major'scale	- comal users -	----------------
----------------	creating'major	theory'menu	- group, usa -	

Guitar Tutorial - Songs

			38 Files	56 Blocks Free:
----------------	b'eyes.ch	land.c.ch	saints.ch	- 6041 monona -
- comal 0.14 -	b'eyes.mel	land.c.mel	saints.mel	- madison, wi -
- program -	bailey.ch	land.ch	----------------	- 53716 -
----------------	bailey.mel	land.mel	-for more info -	----------------
song'menu	down'val.ch	r'sun.ch	- send sase to -	- 608-222-4432 -
----------------	down'val.mel	r'sun.mel	----------------	----------------
- songs -	grace.ch	rrv.ch	- comal users -	
----------------	grace.mel	rrv.mel	- group, usa -	

Guitar Tutorial -Hardcopy

			39 Files	199 Blocks Free:
----------------	----------------	hc.grace.mel	hc.rrv.mel	- group, usa -
- comal 0.14 -	hc.b'eyes.ch	hc.land.c.ch	hc.saints.ch	- 6041 monona -
- program -	hc.b'eyes.mel	hc.land.c.mel	hc.saints.mel	- madison, wi -
----------------	hc.bailey.ch	hc.land.ch	----------------	- 53716 -
print'menu	hc.bailey.mel	hc.land.mel	-for more info -	----------------
----------------	hc.down'val.ch	hc.r'sun.ch	- send sase to -	- 608-222-4432 -
- printer -	hc.down'val.mel	hc.r'sun.mel	----------------	----------------
- programs -	hc.grace.ch	hc.rrv.ch	- comal users -	

Data Base Disk - 0.14

			44 Files	8 Blocks Free:
bootslow	~ comal 0.14 ~	filing	----------------	~ comal users ~
fastboot	~ programs ~	free'form'db	dat.ffdb	~ group, usa ~
c64 comal 0.14	----------------	guess'it	phone.dat	~ 6041 monona ~
comalerrors	1541 database	star'trek'db	ran.doctorwho	~ madison, wi ~
ml.sizzle	boot'data'base	telephone	ran.startrek	~ 53716 ~
hi	boot'tutor	tutor'amnesia	----------------	----------------
menu	data'base'mgr	tutor'remember	~for more info ~	~ 608-222-4432 ~
window-rs232.obj	dbase14	----------------	~ send sase to ~	----------------
----------------	doctorwhodb	~ data files ~	----------------	

Data Base Disk - 2.0

			88 Files	1 Blocks Free:
hi	db'data	~ data base ~	~ text files ~	~ require the ~
menu	db'help.def	----------------	----------------	~ comal 2.0 ~
----------------	db'help.lab	~ len lindsay ~	txt.about comal	~ cartridge to ~
~ data base ~	db'help.rpt	~ and ~	txt.cbase	~ run ~
~ manager ~	db'name	~charl phillips~	txt.db'tutorial	----------------
----------------	----------------	----------------	txt.star trek	~for more info ~
~ robert ~	~ video filer ~	star'trek'db	txt.video filer	~ send sase to ~
~shingledecker ~	~ system ~	ran.startrek	----------------	----------------
----------------	----------------	----------------	~these programs~	~ comal users ~
db'boot	~ bob hoerter ~	~ cbase ~	~must be moved ~	~ group, usa ~
db'define	----------------	----------------	~ onto another ~	~ 6041 monona ~
db'help	videofilersystem	~ russ jensen ~	~ disk before ~	~ madison, wi ~
db'labels	ext.correctfiles	----------------	~ being used ~	~ 53716 ~
db'maintenance	ext.enter'record	cbase	----------------	----------------
db'menu	ext.split'file	sample.f-for	~ all programs ~	~(608)222-4432 ~
db'report	ext.sub-file	sample.topr	~ on this disk ~	----------------
db'sort	----------------	sample.data	~are written in~	
db'squash	~ star trek ~	----------------	~comal 2.0 and ~	

Font Disk #3

			108 Files	14 Blocks Free:
bootslow	font'editor	set.future.a	~ greensboro, ~	~use only. this~
fastboot	----------------	set.grid.a	~ nc 27420 ~	~ disk may not ~
c64 comal 0.14	~ character ~	set.gumby.a	----------------	~be included in~
comalerrors	~ fonts ~	set.gumby.b	~copyright 1985~	~ user group ~
ml.sizzle	----------------	set.halo.a	~ j. blake ~	~ libraries or ~
----------------	font.astronomy	set.halo.b	~ lambert ~	~ uploaded to ~
~ this is a ~	font.buckrogers	set.largetype.a	----------------	~ any bbs ~
~ special hi ~	font.future	set.largetype.b	~ all rights ~	----------------
~ program ~	font.grid	set.nu'deco.a	~ reserved ~	----------------
----------------	font.gumby	set.nu'deco.b	----------------	~for more info ~
~ you must run ~	font.halo	set.small.a	font.boone	~ send sase to ~
~it before you ~	font.largetype	set.squat.a	font.chicago	----------------
~run either of ~	font.nu'deco	set.standard.b	font.newyork	~ comal users ~
~ the font ~	font.small	set.topps.a	font.vilas	~ group, usa ~
~ programs ~	font.squat	----------------	set.boone.b	~ 6041 monona ~
----------------	font.topps	~ these fonts ~	set.chicago.b	~ madison, wi ~
hi	----------------	~ taken with ~	set.newyork.b	~ 53716 ~
----------------	~ character ~	~ permission ~	set.vilas.b	----------------
~ comal 0.14 ~	~ sets ~	~ from ~	----------------	~ 608-222-4432 ~
~ programs ~	----------------	----------------	~ you may make ~	----------------
----------------	set.astronomy.a	~ /speedpak/ ~	~backup copies ~	
demo/load'font	set.buckrogers.a	~ pob 22022 ~	~ for your own ~	

3 Programs in Detail Disk

			62 Files	6 Blocks Free:
hi	----------------	bulletin.board	copy.help	~ send sase to ~
----------------	home'accountant	edit.user.id	log.on.notice	----------------
~ program #1 ~	account'work	sysop.menu	messages.help	~ comal users ~
----------------	init	ext.apply	passwords.help	~ group, usa ~
~ black book ~	main'prog	ext.copytext	user.id.help	~ 6041 monona ~
----------------	posting	ext.editext	----------------	~ madison, wi ~
bb'loader	reports	ext.killtext	~ you may only ~	~ 53716 ~
black'book	----------------	ext.readtext	~make copies of~	----------------
----------------	~ program #3 ~	ext.writext	~this disk for ~	~ 608-222-4432 ~
~ program #2 ~	----------------	----------------	~ your own use ~	----------------
----------------	~bulletin board~	~data files for~	----------------	
~ home ~	----------------	~bulletin board~	----------------	
~ accountant ~	tel-com	----------------	~for more info ~	

MANDELBROT ETC.

Robert Ross sent us this letter, including another update to Mandelbrot:

Please remind cartridge users of two situations where the use of an integer variable is faster:

1) integer FOR control var
2) the :+ / :- variable increment / decrement

In issue #19, the improved PROC mandelbrot could benefit from using a FOR loop. I once calculated a full screen Mandelbrot, writing the iteration counts to disk, and it took over 80 hours. And it seems as if it wasn't real long afterwards that my original CBM power supply died. If you're going to program Mandelbrot, speed is a major factor, somewhere after accuracy. Both of the published PROCs will occasionally do PROC dot twice and both should be examined along with dot for the situation where it = 1: dot is done first with pencolor(0) because of the ">="; after the calculation of sizez, dot is done again with pencolor(0) regardless of the value calculated for sizez because the ">=" still applies.

As for speed, rewriting the PROC using a FOR loop with an integer counter will measurably improve it. In addition to the slower WHILE loop, both of the WHILE test conditions are in effect tested again inside the loop. Using some unnecessary variables and initializing some variables

more»

unnecessarily also takes a bit of extra time, but may have made it easier to write the program originally and may make it easier to revise later. The listing shown below gives two possible rewrites. PROC mrr could replace PROC mandelbrot directly, although the ">=" in dot should be ">" to work properly. (In the listing, dot has been rewritten further.) GOTO is used to exit the FOR loop; if ever a GOTO is written in cartridge COMAL that is better than the other options it will probably be an exit from a structure (LOOP - not in the kernal I have read - does better with the most likely candidate for a GOTO, to create an unending loop, although WHILE TRUE DO serves well in the absence of LOOP and is a bit poetic and can be read as a good rule for living.). In PROC mrr the GOTO could be removed by inserting count:=it+1 as the line before the FOR statement and replacing the IF structure with IF az2+bz2>=4 THEN count:=x#; x#:=it (to save the count value and to get out of the loop). Though I have done it before, to me it seems a bit awkward to have to manipulate the counter to get out of such a loop. PROC mbdot shows a better way for this situation. PROC calculate would call mbdot instead of mandelbrot; mbdot invokes mbrt, which is mandelbrot rewritten to be a function. Two separate RETURN statements, one from inside the FOR loop, return the appropriate values. (It is natural to write mandelbrot as a function, but I might use a GOTO if there were

Sprite Disk #1

			142 Files	156 Blocks Free:
bootslow	target'sighting	imag.cat2	imag.elephant1	imag.koala5
fastboot	view'sprites	imag.cat3	imag.elephant2	imag.koala6
c64 comal 0.14	word'hider	imag.cat4	imag.elephant3	imag.koala7
comalerrors	--sprite images-	imag.cat5	imag.elephant4	imag.koala8
ml.sizzle	imag.'a'	imag.cat6	imag.elephant5	imag.liner
hi	imag.'c'	imag.cat7	imag.elephant6	imag.man'sd
menu	imag.'f'	imag.cat8	imag.elephant7	imag.man1
names.dat	imag.'k'	imag.church3	imag.elephant8	imag.man2
---text files---	imag.'l'	imag.comal today	imag.f18	imag.man3
comal article	imag.'m'	imag.crayon	imag.fish1	imag.man4
graphics.txt	imag.'n'	imag.deer1	imag.fish2	imag.man5
info.txt	imag.'o'	imag.deer2	imag.fish3	imag.man6
keywords.txt	imag.'r'	imag.deer3	imag.fish4	imag.man7
metamorphose.txt	imag.bat1	imag.deer4	imag.fish5	imag.man8
sprite look.txt	imag.bat2	imag.deer5	imag.fish6	imag.men0
sprites.txt	imag.bat3	imag.deer6	imag.fish7	imag.men1
--.14 programs--	imag.bird1	imag.deer7	imag.fish8	imag.men2
add'practice	imag.bird2	imag.deer8	imag.fox	imag.monkey1
box	imag.bird3	imag.dog1	imag.gull1	imag.monkey2
guess'number	imag.bird4	imag.dog2	imag.gull2	imag.monkey3
hopping	imag.bird5	imag.dog3	imag.gull3	imag.monkey4
invisible	imag.bird6	imag.dog4	imag.house	imag.monkey5
metamorphose	imag.bird7	imag.dog5	imag.koala1	imag.monkey6
sprite-sample4	imag.bird8	imag.dog6	imag.koala2	imag.monkey7
sprite'editor62	imag.boat	imag.dog7	imag.koala3	imag.monkey8
target	imag.cat1	imag.dog8	imag.koala4	imag.queen

Sprite Disk #1 - continued

imag.question	imag.santa2	imag.tree4	---procedures---
imag.santa0	imag.tree	imag.tree7	loadshape.proc
imag.santa1	imag.tree'woman	imag.woman'sd	saveshape.proc

Sprite Disk #2

			128 Files	248 Blocks Free:
hi	shap.bat3	shap.deer6	shap.fish7	shap.men1
menu	shap.bird1	shap.deer7	shap.fish8	shap.men2
--2.0 programs--	shap.bird2	shap.deer8	shap.fox	shap.monkey1
all'at'once	shap.bird3	shap.dog1	shap.gull1	shap.monkey2
all'at'once2	shap.bird4	shap.dog2	shap.gull2	shap.monkey3
all'at'once3	shap.bird5	shap.dog3	shap.gull3	shap.monkey4
font'sprite	shap.bird6	shap.dog4	shap.house	shap.monkey5
sprite creator	shap.bird7	shap.dog5	shap.koala1	shap.monkey6
sprite-sample4	shap.bird8	shap.dog6	shap.koala2	shap.monkey7
---text files---	shap.boat	shap.dog7	shap.koala3	shap.monkey8
txt.sprite creat	shap.cat1	shap.dog8	shap.koala4	shap.queen
---2.0 package--	shap.cat2	shap.elephant1	shap.koala5	shap.question
pkg.bits	shap.cat3	shap.elephant2	shap.koala6	shap.santa0
src.bits	shap.cat4	shap.elephant3	shap.koala7	shap.santa1
--sprite shapes-	shap.cat5	shap.elephant4	shap.koala8	shap.santa2
shap.'a'	shap.cat6	shap.elephant5	shap.liner	shap.tree
shap.'c'	shap.cat7	shap.elephant6	shap.man'sd	shap.tree'woman
shap.'f'	shap.cat8	shap.elephant7	shap.man1	shap.tree4
shap.'k'	shap.church3	shap.elephant8	shap.man2	shap.tree7
shap.'l'	shap.comal today	shap.f18	shap.man3	shap.woman'sd
shap.'m'	shap.crayon	shap.fish1	shap.man4	---data files---
shap.'n'	shap.deer1	shap.fish2	shap.man5	dat.bwv779
shap.'o'	shap.deer2	shap.fish3	shap.man6	dat.bwv783
shap.'r'	shap.deer3	shap.fish4	shap.man7	dat.bwv801
shap.bat1	shap.deer4	shap.fish5	shap.man8	
shap.bat2	shap.deer5	shap.fish6	shap.men0	

Graphics Editor

			55 Files	23 Blocks Free:
bootslow	file.1.j	file.12.j	dump.epson	----------------
fastboot	file.2.j	file.13.j	dump.imp	~compacted pix ~
c64 comal 0.14	file.3.j	file.14.j	dump.nec	----------------
comalerrors	file.4.j	file.15.j	dump.nec.b	compact pix
ml.sizzle	file.5.j	file.16.j	dump.oki92	lightbulb.crg
hi	file.6.j	----------------	dump.oliv	objects.crg
----------------	file.7.j	- screen dumps -	----------------	----------------
-support files -	file.8.j	----------------	- bitmap pix -	- graphics -
----------------	file.9.j	dump.1520	----------------	- editor by -
- do not load -	file.10.j	dump.1525	directory	~colin thompson~
----------------	file.11.j	dump.bx80	calvin.hrg	----------------

Today 20/Special

			99 Files	1 Blocks Free:
bootslow	program'outliner	graph'fp'err-2.0	proc.longdiv	~today disk 20 ~
fastboot	-----------------	mod'tutorial	proc.longenviron	~ and special ~
c64 comal 0.14	~compiled 0.14 ~	outliner-2.0	proc.longmul	~ edition disk ~
comalerrors	~ program ~	rod	-----------------	-----------------
ml.sizzle	-----------------	xmashayes/barton	~support files ~	~ do not give ~
hi	~ load and run ~	xmassharp/taylor	~ for rod the ~	~out copies of ~
menu	~ from basic ~	-----------------	~ roadman ~	~ this disk ~
names.dat	-----------------	~ functions ~	-----------------	-----------------
-----------------	shredder	-----------------	problems	~for more info ~
~ text files ~	-----------------	func.position'in	roderigue	~ send sase to ~
-----------------	~ comal 2.0 ~	func.write'prote	shap.down'rod	-----------------
keywords.txt	~ programs ~	str.proc	shap.lt'rod	~ comal users ~
graphics.txt	-----------------	val.func	shap.rt'rod	~ group, usa ~
sprites.txt	3d'fractals	-----------------	shap.up'rod	~ 6041 monona ~
-----------------	calculate'pi-2.0	~ procedures ~	-----------------	~ madison, wi ~
~ comal 0.14 ~	chip.unerase	-----------------	~ packages ~	~ 53716 ~
~ programs ~	copy/compare	proc.environ	-----------------	-----------------
-----------------	correct'disk2	proc.exactmul2	pkg.text	~ 608-222-4432 ~
calculate'pi	demo/text2	proc.longabs	src.text	-----------------
graph'fp'error	dot'images	proc.longadd2	-----------------	

no other reason to put the FOR in a separate function when I needed the counter value.) As written here, zero is returned instead of it+1 when all iterations are done with az2+bz2 less than 4; this perhaps simplifies the rest of mbdot. Each of these rewrites is faster than using the published versions; using mbdot is slightly faster than mrr. The listing provided by Robert Ross is shown below to illustrate points raised in his letter:

```
ZONE 5 //Mandelbrot 2.0 Revisited Revised
it:=50 //by Robert Ross
//           see notes in letter above
PROC mbdot
  TIME 0
  count:=mbrt
  IF count THEN count:=count MOD 3+6
  pencolor(count)
  plot(ac,bc)
  PRINT TIME,count,
ENDPROC mbdot
//
FUNC mbrt
  az2:=0; bz2:=0 // ;az:=0; bz:=0
  FOR x#:=1 TO it DO
    bz:=2*az*bz+bc; az:=az2-bz2+ac
    az2:=az*az; bz2:=bz*bz
    IF az2+bz2>=4 THEN RETURN x#
  ENDFOR x#
  RETURN 0
ENDFUNC mbrt
//
PROC mrr
  TIME 0
  az2:=0; bz2:=0
  FOR x#:=1 TO it DO
    bz:=2*az*bz+bc; az:=az2-bz2+ac
    az2:=az*az; bz2:=bz*bz
    IF az2+bz2>=4 THEN
      count:=x#
      GOTO dodot
    ENDIF
  ENDFOR x#
  count:=it+1
dodot:
  dot
  PRINT TIME,count,
ENDPROC mrr
//
FOR bc:=-.915 TO -.865 STEP 1e-03 DO
  FOR ac:=-.1525 TO -.1025 STEP 2e-03 DO
    PRINT bc,ac;":"13"     ",
    az:=0; bz:=0
    mbdot
    az:=0; bz:=0
    mrr
    PRINT
    PRINT
  ENDFOR ac
ENDFOR bc
//
PROC dot
  IF count>it THEN
    pencolor(0)
  ELSE
    pencolor(count MOD 3+6)
  ENDIF
  plot(ac,bc)
ENDPROC dot
//
//  Redefine PENCOLOR and PLOT:
PROC pencolor(z)
  pc:=z
ENDPROC pencolor
//
PROC plot(a,b)
  PRINT pc,
ENDPROC plot
```

ORDER FORM Subscriber#__________

Name:______________________________

Street:______________________________

City/St/Zip:__________________________

Visa/MC#:____________________________

Exp Date:______ Signature:________________

Dec '87-Prices subject to change without notice

TO ORDER:

- Pull out this section; fill in above info
- Check [x] each item you want to order
- Add up total for items (fill in below)
- Add shipping/handling (fill in below)
- Send check/money order for Grand Total OR fill in charge info (above) ... and we will calculate the total & shipping
- Mail in the order... or call 608-222-4432

SUBSCRIPTIONS:

Expired subscribers must renew before they may order at subscriber prices (renewal starts with the issue where you left off). New subscribers can order at the same time as subscribing.

[ ] $12.95 - 4 issue subscription/renewal
(Canada/APO add $1 per issue, 1st Class)
[ ] $29.95 - 4 disk subscription/renewal

[ ] $3.95 each backissues; circle issues wanted:
5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
[ ] $3.95 each **Early Issues**: 1 2 4
[ ] $16.95 *COMAL Yesterday* (spiral bound #1-4)
(plus $3 shipping)

NOTICE: Minimum order $10. Shipping is extra: $3 minimum shipping; $3 per book; Canada, APO & 1st Class add $1 more per book and newsletter issue. Newsletter is published about 4 times per year; size and format may vary. Subscriptions may not extend more than 6 issues past the current issue. No money back for cancelled disk or newsletter subscriptions; however, for each future issue cancelled, one backissue (above) or one disk (from the disk column) may be chosen as compensation. All orders must be prepaid in US Dollars; Canada and USA only. Prices shown are subscriber prices and reflect a $2 discount. Allow 2 weeks for checks to clear. $10 charge for checks not honored by the bank. Wisconsin residents add 5% sales tax.

ENTER TOTALS HERE:

Item Total ($10 minimum):$__________
Ship Total ($3 minimum): $__________
Grand Total enclosed: $__________

SYSTEMS:

[ ] **C64 Power Box**db
Includes 3 utility disks, a toolbox of about 250 procedures and functions, and a free compiler. $29.95 + $4 ship
[ ] **Starter Kit Option**: 12 issues of COMAL Today, 56 page index, 2 books and 5 disks! Over 1,000 pages! $29.95 + $4 ship
[ ] **C64 Keyboard Overlay Option**: excellent condensed command reference. $2.50
[ ] **Option** ... $6.95 Doc Box

[ ] **C64 COMAL 2.0 Cart Complete***
This is the way to go! It includes tutorials, references, packages, and superchip on disk. Cartridge with all 4 options: $158.95+$7 ship
[ ] **C64 COMAL 2.0 Cartridge*** (34 in stock)
64K cartridge with empty socket for up to 32K EPROM. Plain, no documentation $89.95
[ ] **Deluxe Option**db: 3 books: *Cart Keywords*, *Cart Graphics & Sound*, *Common COMAL Reference*; 4 cart demo disks. $29.95+$4 ship
[ ] **Packages Option**db: 4 books: *COMAL 2.0 Packages*, *Packages Library 1* (17 packages), *Packages Library 2* (24 packages); Superchip on Disk/Notes (9 packages) $29.95 + $4 ship
[ ] **Applications/Tutorial Option**db: three books: *COMAL Collage*, *3 Programs In Detail*, *Graph paper*. $29.95 + $4 ship
[ ] **Option** ... $6.95 Doc Box

[ ] **Super Chip** plugs into C64 cart - $24.95*

[ ] **CP/M COMAL 2.10**
Full COMAL system disk plus the DEMO disk, packed in a Doc Box with manual. Works in C128 CP/M mode. $49.95 + $4 ship
[ ] **Compiler Option**: RUNTIME system ... $9.95
[ ] **C128 Graphics Option**: Package disk $9.95
[ ] **CP/M Package Guide Option**: Reference on how to write packages $10.95 + $2 ship
[ ] **Option**: Common COMAL Reference $14.95

[ ] **UniComal IBM PC COMAL 2.1***
Full fast system, with extensive reference & tutorial packed in a Doc Box. $395 +$5 ship
[ ] **Compiler Option**: (PLUS version) adds a Runtime compiler and the Communication Package, with its own manuals packed in its own Doc Box. $195.00 + $4 ship
[ ] **Option**: Common COMAL Reference $14.95

[ ] **C128 COMAL 2.0 Cart***
Special order price: $195.95. This cart works on the C128 in its native mode. ($2 ship)
[ ] **Option**db: Common COMAL Reference $14.95
[ ] **Option** ... $6.95 Doc Box

db=Doc Box pages, requires a Doc Box for use.
*=subject to customs/ship variations/availability.

BOOKS: ($3 shipping each; most cost just $10.95)

[ ] $3.95 **Cartridge Keywords**db
36 pages. Lists all the keywords built into the cartridge in alphabetic order complete with syntax and example. Also includes all keywords in the 11 built in packages.

[ ] $14.95 **Common COMAL Reference**db
238 page detailed cross reference to the COMAL implementations in the USA by Len Lindsay (formerly *COMAL Cross Reference*) Covers: C64 COMAL 2.0, C128 COMAL 2.0, CP/M COMAL 2.10, Mytech IBM PC COMAL 2.0, and UniComal IBM PC COMAL 2.1
[ ] **Cartridge Keywords Option - $3.95**
[ ] **Cartridge Graphics & Sound Option - $4.50**

[ ] $10.95 **Beginning COMAL**¤
#8 all time best seller by Borge Christensen
333 pages - General Beginners Textbook, elementary school level, written by the founder of COMAL. This book is an easy reading text. You should find Borge has a good writing style, with a definite European flair.
[ ] **Matching C64 disk option - $7.95**

[ ] $10.95 **Foundations with COMAL**¤
#10 all time best seller by John Kelly
363 pages - General Beginners Textbook Jr/Sr High School level, including a section on C64 turtle graphics. A good text for those serious about programming.
[ ] **Matching C64 disk option - $7.95**

[ ] $3.95 **COMAL From A to Z**¤
#1 all time best seller by Borge Christensen
64 pages - Mini 0.14 Reference book
Written by the founder of COMAL

[ ] $3.95 **COMAL Workbook**¤
#4 all time best seller by Gordon Shigley
69 pages - 0.14 Tutorial Workbook
Companion to the Tutorial Disk, great for beginners, full sized fill in the blank style.
[ ] **Tutorial Disk Option**: $7.95

[ ] $3.95 **Index to COMAL Today 1-12**
#3 best seller for Nov 87 by Kevin Quiggle
52 page, 4,848 entry index to COMAL Today. The back issues of *COMAL Today* are a treasure trove of COMAL information and programs! This index is your key to #1-12.
[ ] **Index Disk Option**: $7.95

[ ] $10.95 **CP/M COMAL Package Guide**db
76 pages - advanced package reference
The guide to making your own packages for CP/M COMAL by Richard Bain

[ ] $10.95 **Library of Functions & Procs**db
#1 best seller for Nov 87.
80 pages with disk by Kevin Quiggle
Over 100 procedures & functions for 0.14!
Part of Power Box.

[ ] $4.50 **Cartridge Graphics & Sound**¤db
#6 all time best seller by Captain COMAL
64 pages - 2.0 packages reference
Reference guide to all the commands in the 11 built in cartridge packages.

[ ] $10.95 **COMAL 2.0 Packages**db
#1 best seller for Oct 87 by Jesse Knight
108 pages with disk - package reference
How to write a package in Machine Code; includes C64 comsymb & supermon. For advanced users.

[ ] $10.95 **Package Library Vol 1**db
compiled by David Stidolph
76 pages with disk - package collection
17 packages ready to use, many with source code, plus the Smooth Scroll Editor!

[ ] $10.95 **Package Library Vol 2**db
#5 best seller for Nov 87.
67 pages with disk - package collection
24 example packages ready to use, most with source code, plus Disassembler, Re-Linker, De-Linker, Package Maker, Package Lister, and more.

[ ] $10.95 **COMAL Collage**db
#5 best seller for Oct, 168 pages with disk
2.0 programming guide by Frank & Melody Tymon; including a graphics and sprites tutorial with full sized example programs.

[ ] $10.95 **3 Programs in Detail**db
82 pages with disk by Doug Bittinger
Three 2.0 application programs explained: Blackbook (name/address system), Home Accountant, and BBS.

[ ] $10.95 **Graph Paper**db
#4 best seller for Aug, 52 pages with disk by Garrett Hughes; 2.0 Function graphing system. The unLISTable program includes a version for the Commodore Mouse.

[ ] $10.95 **COMAL Quick / Utility 2 & 3**db
#2 best seller for Nov 87.
20 pages with two disks by Jesse Knight
Fast loading COMAL 0.14, printer programs, and other utility programs.
Part of Power Box.

db = Doc Box pages
¤ = while supplies last (out of print)